A Hybrid Error Control Scheme for ATM Networks

by

Hazem Helmi Selmi

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

ELECTRICAL ENGINEERING

May, 2000

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

A HYBRID ERROR CONTROL SCHEME FOR ATM NETWORKS

BY

HAZEM HELMI SELMI

A Thesis Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

ELECTRICAL ENGINEERING

MAY 2000

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **Hazem Helmi Selmi** under the direction of his Thesis Advisor and approved by his Thesis Committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE in ELECTRICAL ENGINEERING.**

Thesis committee

Dr. Maan A. Kousa (**Chairman**)

Dr. Mahmoud Dawoud (**Member**)

Dr. Saud Al-Semari (**Member**)

Dr. Adil Balghonaim (**Member**)

Dr. Samir Al-Baiyat
Chairman, Electrical Engineering

Dr. Abdallah Al-Shehri
Dean, Graduate Studies

10/6/2002
Date

To my mother and my father

ACKNOWLEDGMENT

Acknowledgment is due to King Fahd University of Petroleum and Minerals for support of this work.

I would like to express my appreciation to Dr. Maan Kousa, my principle advisor, for his careful guidance and encouragement through this research.

I am also grateful to other committee members Dr. M. Dawoud, Dr. Adel Balghonaim and Dr. Saud Al-Semari for their useful and valuable suggestions.

Thanks are also due to all friends who helped and/or encouraged my during the work in this thesis. Among all, many thanks to Saib Al-Haddad and Ahmed Fathi.

Last but not least, thanks are due to all my family members for their patience, support and sacrifice. Special thanks to Um-Sohaib, Sohaib, Mariam, Sara and Osama.

خلاصة الرسالة

اسم الطالب : حازم حلمي سلمي

عنوان الرسالة : نظام مدجن للتحكم في الأخطاء لشبكات ATM

التخصص : الهندسة الكهربائية

إن تقنية طور النقل المتزامن ATM يتوقع لها القيام بدور أساسي في التطور المستقبلي لشبكات الاتصال العالمية التي تقدم خدمات مختلفة بطريقة مدمجة. تعاني شبكات ATM من فقدان الخلايا المرسلة خصوصاً إذا كانت القناة مشوشة. في هذه الحالة لابد من استخدام أنظمة للتحكم في الأخطاء لاستعادة الخلايا المفقودة. هذا البحث قام بدراسة نظام لاستعادة الخلايا المفقودة يعتمد على شفرة ثنائية الأبعاد أطلق عليه اختصاراً اسم (PCPC) . قدم البحث دراسة جديدة للأشكال المحتملة لمجموعة من الخلايا المفقودة في الشفرات ثنائية الأبعاد، وتم تحديد معدل الخلايا المفقودة بعد استخدام PCPC. وقام البحث بدراسة استخدام ARQ مع PCPC باستخدام عدة طرق لإعادة الإرسال.

وصلت الدراسة إلى أن الدراسة الجديدة للأشكال المحتملة لمجموعة من الخلايا المفقودة في الشفرات ثنائية الأبعاد مفيدة جداً في تحديد ما إذا كانت مجموعة من الخلايا المفقودة قابلة للاستعادة أم غير قابلة، وأن التحديد المستنتج لمعدل الخلايا المفقودة بعد استخدام PCPC دقيق جداً ، وأن PCPC لديه قدرة قوية على استعادة الخلايا المفقودة. أما ما يمكن أن يضيفه استخدام ARQ مع PCPC فيعتمد على طريقة إعادة إرسال الخلايا.

درجة الماجستير في العلوم

جامعة الملك فهد للبترول والمعادن

الظهران ـ المملكة العربية السعودية

صفر ١٤٢١

ABSTRACT

Name: Hazem Helmi Selmi

A HYBRID ERROR CONTROL SCHEME FOR ATM NETWORKS.

Major Field : Electrical Engineering

May 2000

Asynchronous transfer mode (ATM) technology is expected to have an important role in the future evolution of global integrated services communication networks. ATM networks suffer from cell loss. The problem becomes more sever when part of the connection is noisy. In this case error control schemes must be incorporated to recover lost cells. A cell loss recovery scheme for operation over a noisy ATM scheme is analyzed. This scheme is based on the parity check product code (PCPC). A novel algorithm for analyzing the structure of the lost cell patterns in product codes is presented. An upper bound on post decoding cell loss rate in the PCPC is presented. The use of an ARQ scheme on the top of PCPC is also investigated with various retransmission strategies.

It was found that the proposed novel algorithm of structural analysis is very useful in identifying the recoverable cell loss patterns, the derived bound on post-decoding cell loss rate is almost exact, and the PCPC is very powerful in recovering lost cells. The advantage of the hybrid ARQ/PCPC over the pure ARQ depends on the retransmission strategy.

MASTER OF SCIENCE

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS
DHAHRAN, SAUDI ARABIA

May 2000

TABLE OF CONTENTS

CHAPTER 2

A Novel Structural Analysis of Lost Cell Patterns in Product Codes

CHAPTER 3

Performance Bounds for Post Decoding Cell loss Rate in PCPC

CHAPTER 4

Hybrid ARQ/PCPC for ATM networks

CHAPTER 5

Appendix A

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

AAL : ATM Adaptation Later.

ACK : Positive Acknowledgment.

ARQ : Automatic Repeat Request.

ATM : Asynchronous Transfer Mode.

B-ISDN : Broadband Integrated Services Digital Network.

BER : Bit Error Rate.

CLR : Cell Loss Rate.

CRC : Cyclic Redundancy Check.

FEC : Forward Error control Scheme

i : Number of Lost Cells

ISDN : Integrated Service Digital Networks.

k/n : Code Rate.

NACK : Negative Acknowledgment.

PCPC : Single Parity Check Product Code

QoS : Quality of Servies.

RBP : Recoverable Basic Pattern.

R_i : Total of Definitely Recoverable Patterns of size i cells.

RS : Reed-Solomon Codes.

UBP : Unrecoverable Basic Patterns.

U_i : Total of unrecoverable patterns of size i cells.

V_i : Recoverable Patterns Generated from RBPs.

W_i : Recoverable Patterns Generated from UBPs.

CHAPTER 1

INTRODUCTION

The title of this thesis is hybrid *error control* in *ATM* networks. In order to make it easy for the reader to follow the material of this thesis, we devote this Chapter to review the related subjects of error control and ATM networks. At the end of the Chapter, a literature survey of related works and an overview of the proposed work are given.

1.1 Introduction

In 1948, C. Shannon published two papers titled " A Mathematical Theory of Communication". These classic papers formulated some probabilistic models to describe the reliable digital communication and its limits [1]. What Shannon

introduced in those papers is called "information theory" even-though this term was never used in his papers [2]. Introducing "information theory" is considered to be the real theoretical foundation of error-control coding and consequently digital communication in general. Although sampling theory (studied by Nyquist 1928) and some other digital communication aspects are older than Shannon work, he is still considered to be the father of error control in digital communications. Researchers of digital communications believed that his efforts established the mathematical foundations which allows the later developments of this subject [1,2].

The first commercial digital (voice) system was installed in Chicago in 1962. This is the well-known T1 system, which was originally designed to accommodate 24 voice channels. These channels cover short distances in metropolitan areas. The T1 system drew strong research attention, which lead to continuous improvements until it became widely used in USA, Canada and Japan. This system starts the shift to digital communications in practice [2].

Since the research in this thesis falls in the area of **error control** in **ATM networks**, this chapter provides an overview of these two subjects. A general overview of channel coding and error control techniques is given in Section 1.2. As introduction for ATM networks, some important concepts in data networks and the motivations for ATM networks are summarized in Section 1.3. A summery of some aspects in ATM networks is given in the Sub-Section 1.3.3. The aspects discussed for ATM networks are those which are related to our study in this thesis. A literature survey is provided in Section 1.4. A detailed statement of the problem addressed in this thesis is furnished in Section 1.5.

1.2 Channel Coding

Prior to Shannon's information theory, it was believed that to have a reliable digital transmission it is necessary to use very low transmission rate [3]. Or by nowadays terminology, digital communication with acceptable reliability would not be possible without scarifying the throughput. Shannon proved that this is not true. He showed that not only reliable but almost error free communication is possible with any transmission rate (throughput) below the channel capacity by choosing a proper code [1]. Even though Shannon theory did not explain how to find the desired code, it draw the attention towards channel coding. The first non trivial error-control code was introduced by Hamming in 1950. That was two years after Shannon introduced the "information theory". Theoretical and mathematical researches and developments of coding continued extensively in the subsequent years of the 50's and 60's. In those years, Cyclic codes theories, BCH codes, RS codes, Convolutional codes, Viterbi decoding algorithm and many of the other famous theories, codes, decoding algorithms were developed. However, it was not until the 70's when error-control codes were implemented in practical and commercial digital communications systems.

It is known that Bit Error Rate (*BER*) is a function of the signal to noise ratio (E_b/N_o) and the modulation scheme used. Without error control coding, the desired *BER* for a specific channel can be achieved either by increasing the transmitted power of the signal or using a more powerful modulation scheme. Both solutions are costly. By using error control, the same *BER* can be achieved with less transmitted power. Or the same E_b/N_o can produce a better *BER*.

Channel coding is used to improve the reliability of the data delivered to the user. The position of channel encoder (and decoder) in a general digital communication block diagram is shown in Figure 1.1. Discreet data in this figure can be a bit stream from a digital source or of an analogue to digital converter.

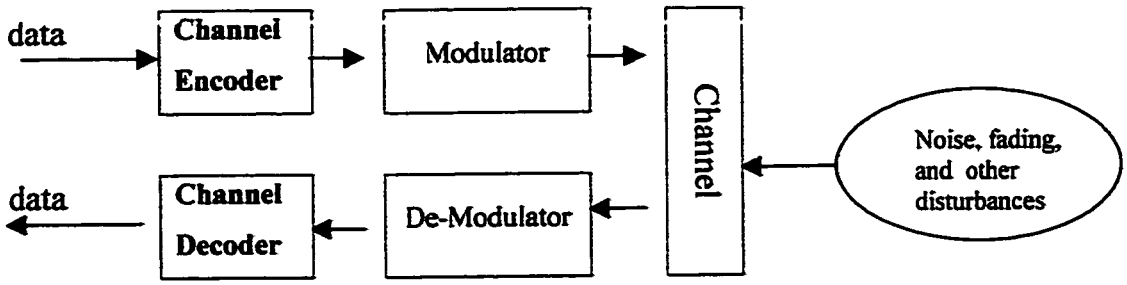


Figure 1.1 : A general digital communication block diagram.

The redundancy added by channel coding is used for *Error Detection* and/or *Error correction*. In some special cases this redundancy may be used in a different way called *Erasers Decoding*. The first two techniques are explained briefly in the next subsection, while Erasures decoding is discussed in Subsection 1.2.4.

1.2.1 Error correction and detection:

The received data may contain some errors due the channel imperfections. In the receiver, the decoder tries to identify any invalid codeword. Any valid codeword is assumed to be correct. This process is called *error detection*. The decoder will fail in error detection if and only if the channel modified the transmitted word to

another valid word [4]. Alternatively, in *error correction*, the decoder may attempt to correct some of the received words. Error detection and error correction are achieved using the redundancy added by the encoder. The redundancy bits are added according to the rules of the used code. Any code is characterized by the Hamming distance d. A code with Hamming distance d can detect up to d-1 errors when used for error detection only, or correct up to $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors when used for error correction only. In general a code with minimum distance d can correct up to t errors and simultaneously detect up to λ more errors such that $d = 2t + \lambda + 1$.

1.2.2 Error Control Schemes:

There are two basic techniques of error control, namely Automatic Repeat Request (ARQ) and Foreword Error Correction (FEC). In the ARQ, if no errors are detected by the decoder, the original information bits are delivered to the user. Otherwise, the receiver discards the received word and sends back a negative acknowledgment (NACK) requesting for a retransmission of the same codeword. Theoretically this process is repeated until the codeword is successfully received. Obviously, in any ARQ system the availability of a feedback channel is a must. From the above, it is clear that coding is used for the sole purpose of error detection. That is why high rate error detecting codes are used in ARQ systems.

There are three basic protocols of retransmission in an ARQ system: stop-and wait (SW), go-back-N (GBN) and selective repeat (SR). In SW-ARQ the transmitter stops after each transmission and remains idle waiting for acknowledgment to be received back. If positive acknowledgment is received (ACK), a new cell is transmitted, while a negative acknowledgment (NACK)

means re-transmitting the same cell. In GBN-ARQ and SR-ARQ the transmitter is continuously transmitting cells until it receives a NACK. In this case, GBN system goes back to the negatively acknowledged cell and retransmit it with all the successive cells. In SR-ARQ, only the negatively acknowledged cells are retransmitted.

In FEC systems the redundancy added by channel coding is used by the decoder for error correction of the detected invalid codeword. When the decoder detects the presence of errors in a received word it attempts to locate and correct them. The decoded word is then delivered to the user.

The performance of digital communication systems are usually evaluated by their "throughput" and "reliability". Throughput is known by the ratio of useful information bits accepted by the receiver to the total transmitted bits per unit time. Reliability is a measure of the correctness of the received data. Both error control techniques will enhance the reliability of communication system at the cost of the throughput.

In FEC the decoded word must be delivered to the user regardless of whether it is correct or not. This implies that the reliability of systems using FEC technique is less than those using ARQ systems specially for poor channels. As a matter of fact, ARQ can give almost the same reliability as that of an ideal channel. The only degradation of reliability from the ideal case comes from the possibility that an erroneous word passes undetected. This happens when the errors in a codeworde are beyond the detection capabilities of the code used. The probability of this event is very low in relatively good channels especially when a strong error detecting code is used

Since there are no retransmissions in a FEC system, it provides constant throughput efficiency, set by the code rate, regardless of the channel condition. In an ARQ system, the process of retransmission degrade the throughput efficiency specially for poor channels.

In summery, both techniques have relative advantages and disadvantages. However, ARQ is usually preferred for data communication systems [3]. When no feedback is available, as in data storage, FEC becomes the only choice. A combination system that has the advantages of both ARQ and FEC systems is termed in the literature *Hybrid ARQ*. In such systems an FEC is used in conjunction with ARQ trying to enhance the reliability with minimum degradation of the throughput

In hybrid ARQ, an error detection code is used for ARQ and another code is used for FEC. At the receiver, the decoder first attempts to correct any error in the received codeword using the error correction code. Then the error detection code tries to detect any error in codeword received from the previous stage. If the decoder fails to detect an error, it assumes that it is a correct codeword and it is delivered to the user. Otherwise if an error is detected the receiver discards the codeword and requests a retransmission of the same codeword.

In some cases, one code only is used for error correction and the erroneously received words which are beyond its correction capability are requested to be retransmitted. This scheme simplifies the encoding and decoding process and is still called hybrid ARQ [3,4].

1.2.3 Error control codes:

There are many types of codes that can be used in channel coding. Examples of the most famous types of codes are parity check codes, Hamming codes, CRC codes, BCH codes and RS codes. Those codes are all from one family called block codes. Another family of codes is the convolutional codes. Component codes can be used to enhance the error correction capability as in product codes, concatenated codes and turbo codes. Most of the codes can be used for error detection and/or error correction.

In the remaining of this section we will review the codes and coding schemes that will be encountered in this work, namely: parity check codes, CRC codes, RS codes and product codes.

1.2.3.1 Parity check codes:

In single parity check codes, the encoder adds one bit to each packet of information bits to make the modulo-2 sum of all the bits equal to 0. This is called sometimes even parity check to differentiate it from the odd parity check when the modulo-2 sum of the bits is made equal to 1. However, the later code does not have the all-zero codeword which makes it a nonlinear code. Because of the nice features of linear codes, even parity check is always used and is therefore the default of parity check codes. The parity check codes are error detecting codes; they are able to detect any odd number of errors.

Due to their simplicity, parity check codes are widely used in many practical applications. S. Wicker supported the claim that this code is the most frequently used over the past 50 years [4]. The most famous application is in computer data buses and memories. In some error control systems parity check codes are incorporated in more complicated systems to improve the overall performance. Examples of these systems can be found in [14-19].

1.2.3.2 CRC codes:

The Cyclic Redundancy Codes (CRC codes) are extremely well suited and widely used for error detection only [1, 4]. That is why they are some times defined as high speed error detecting codes [4]. CRC codes are very popular due their powerful detection capability and the simplicity of implementing encoders and decoders. CRC codes are widely used in computer industry [3]. The following is a list of the famous CRC codes with their generator polynomials [4]:

CRC CODE	GENERATOR POLYNOMIAL
CRC-4	$g_4(x) = x^4 + x^3 + x^2 + x + 1$
CRC-7	$g_7(x) = x^7 + x^6 + x^4 + 1 = (x^4 + x^3 + 1)(x^2 + x + 1)(x + 1)$
CRC-8	$g_8(x) = (x^5 + x^4 + x^3 + x^2 + 1)(x^2 + x + 1)(x + 1)$
CRC-12	$g_{12}(x) = x^{12} + x^{11} + x^3 + x^2 + x + 1 = (x^{11} + x^2 + 1)(x + 1)$
CRC-ANSI	$g_{ANSI}(x) = x^{16} + x^{15} + x^2 + 1 = (x^{15} + x + 1)(x + 1)$
CRC-CCITT	$g_{CCITT}(x) = x^{16} + x^{12} + x^5 + 1$ $= (x^{15} + x^{14} + x^{13} + x^{12} + x^4 + x^3 + x^2 + x + 1)(x + 1)$
CRC-SDLC	$g_{SDLC}(x) = x^{16} + x^{15} + x^{13} + x^7 + x^4 + x^2 + x + 1$ $= (x^{14} + x^{13} + x^{12} + x^{10} + x^8 + x^6 + x^5 + x^4 + x^3 + x + 1)$ $\cdot (x + 1)^2$
CRC-24	$g_{24}(x) = x^{24} + x^{23} + x^{14} + x^{12} + x^8 + 1$ $= (x^{10} + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x + 1)$ $\cdot (x^{10} + x^9 + x^6 + x^4 + 1)(x^3 + x^2 + 1)(x + 1)$
CRC-32$_A$	$x^{32} + x^{30} + x^{22} + x^{15} + x^{12} + x^{11} + x^7 + x^6 + x^5 + x$ $= (x^{10} + x^9 + x^8 + x^6 + x^2 + x + 1)(x^{10} + x^7 + x^6 + x^3 + 1)$ $\cdot (x^{10} + x^8 + x^5 + x^4 + 1)(x + 1)(x)$
CRC-32$_B$	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5$ $+ x^4 + x^2 + x + 1$

When an (n,k) CRC code is used for error detection only, they are powerful enough to detect all the following types of errors [1],[4]:

All combinations of d_{min}-1 random errors or less.
All error bursts of length $\leq (n-k)$.
$1-2^{1+k-n}$ of error bursts of length $n-k+1$.
$1-2^{k-n}$ of error bursts of lengths $> n-k+1$.
All the error patterns of an odd number of errors, provided that $g(x)$ has an even number of nonzero coefficients.

Using the above points makes it easy to evaluate the performance of a CRC code. The CCIT CRC has a generator polynomial of degree 16 as shown in the above list and $d_{min}=4$ [5]. Applying the above points results in the following error detection capabilities for the CCIT code:

all combinations of random errors less than 4.
all error bursts of length ≤ 17.
99.997% of error bursts of length 17.
99.9985 % of error bursts of lengths > 17.
All odd weight errors.

1.2.3.3 RS codes:

The Reed-Solomon (RS) codes were invented in 1960 and are named in honor of their inventors. RS codes are considered as a subclass of the none-binary BCH

codes. They are linear, cyclic and perfect codes that are very suitable for burst error correcting and coding with M-ary modulation. RS codes prove to be very powerful, efficiently decodable and consequently widely used in different applications. The most popular application of RS codes is their use digital audio disc [4]. RS codes were used by NASA in the *Voyager* mission to Uranus [5].

One of the nice features of RS codes is that their desired Hamming distance (d_{min}) can be exactly achieved. In other words, an *e*-error correcting code can be designed with d_{min} exactly equal to $2e+1$ or $n-k+1$. RS codes can correct $\left\lfloor \frac{n-k}{2} \right\rfloor$ errors or fill $n-k$ erasures.

1.2.3.4 Product codes:

A product code is a code which uses some simple codes to build a relatively more complex and stronger code. Product codes are achieved by arranging the information bits in a two dimensional array. Then the rows are encoded using some code (C1) and the columns are encoded using the same or a different code (C2). In general, product codes are much easier to decode than the none-product codes of the same size and rate. The general shape of product codes is shown in Figure 1.2.

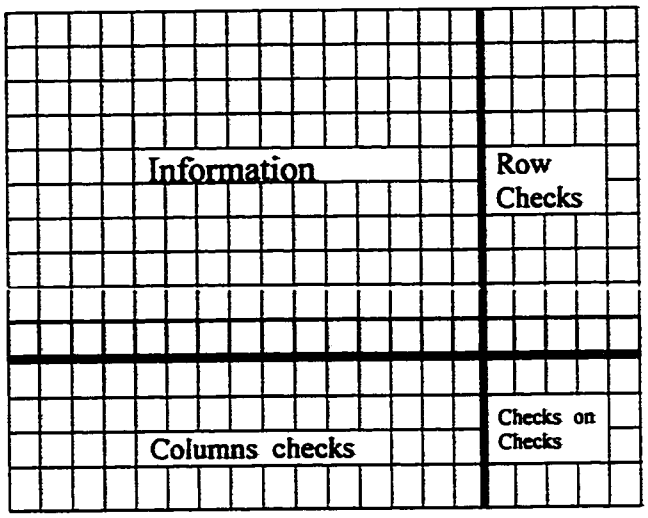


Figure 1.2 General shape of product codes.

It should be noted that each unit in Figure 1.2 could be a bit or a vector of bits. By making all the checks of columns and rows to be simply the single parity checks of the corresponding row or column, we generate a two dimensional **Parity Check Product Code (PCPC).**

The code under study in this work is the PCPC described above. Each unit of the code is called cell. This code is constructed by arranging the cells to be encoded in a matrix. Those cells are encoded by terminating each column and each row by a parity check cell. The parity check cell at the end of each row is obtained by bit–by-bit, mod-2 addition of the cells in that row, while the parity check cell at the end of each column is obtained by mod-2 addition of the cells in that column. It is easy to note that in this easy encoding scheme it makes no difference which direction of encoding is performed first.

Example 1. 1

For a matrix of size 5×6 and cells of 4 bits each, the PCPC will look like:

1011	1000	1100	0000	1110	1010	1011
1111	1111	0011	0011	0000	1000	1000
0001	0011	1010	0101	0101	1111	0111
1100	0000	1000	0101	0100	0110	0011
0011	0011	0001	1011	1100	0011	0110
1010	0111	1100	1001	0011	1010	0001

Figure 1.3 Example of a single parity product code.

Decoding of the PCPC code for lost cell recovery is fairly simple and falls in the area of erasures deeding described in Section 1.4.4. The decoder starts by scanning the first row. If one cell is lost it will be recovered by mod-2 addition of all the cells in that row. If more than one cell is lost, the decoder will not perform any recovery job. The decoder continues scanning all the successive rows and acts in the same way described above. The same decoding algorithm is applied to all the columns. More decoding rounds may be carried out.

1.2.4 Erasures Decoding:

In all of the above discussion we have assumed that the channel has introduced errors in the received words; that is some bits have been inverted. There are cases

were some of the transmitted bits or complete packets are missing or erased from the received data. In theses cases the code must performs *erasure decoding*.

Erasures can take place for different reasons. For example in soft decision decoding, erasures are introduced when the received energy level is too low for the decoder to make reliable decision; hence the bit is erased. Decoders, sometimes, intentionally introduce erasers to simplify decoding. An example is found in some decoding algorithms of product codes. In some network protocols complete packets may be discarded, lost or erased. ATM network is a typical example. An error in the header causes the cell to be discarded. A cell may also be discarded due to buffer overflow.

For any code, its ability to fill erasures is double its ability to correct errors [5]. Therefore, a code with Hamming distance d can fill d-1 erasures if used for erasure decoding only. If the code is used for erasures filling and error correcting simultaneously, then it can fill u erasures and correct e errors such that $d = 2e + u + 1$ [5].

1.3 Data Networks:

Traditionally, Data networks are classified into Local Area Networks (LANs) and Wide Area Networks (WANs). Like voice networks, large distance between users in WANs prohibits the full connectivity between all users and thus necessitates the use of switching techniques. Data in WANs is transferred between source and destination by routing (switching) the packets through a number of intermediate switching nodes.

In LANs, transferring the data between the users takes place directly without the need of any switching technique. The most famous technologies in LANs are the Ethernet and Token Ring. In the mid nineties almost all local networks in the world were using these technologies [6]. The dominant transfer speed of the Ethernet continued to be 10 Mbits/s for long period of time until the mid nineties. Shortly after then Ethernet transfer rates jump to 100 Mbits/s. Nowadays *Gigabit Ethernet* are in use.

1.3.1 Switching:

Circuit switching and *packet switching* are two switching techniques used in digital communication. The concept of circuit switching was known in the analog telephone networks. The first digital circuit switching is the T1 system mentioned before. In circuit-switching technique the two terminals (users) are connected via a physical path or circuit for the whole communication period. Once a circuit is established it remains in place until the call is over. Each circuit has a fixed bandwidth that is dedicated to the two users for the whole communication period. Even-though several circuits can be multiplexed onto a link, there is still a significant waste of bandwidth in circuit switching. Yet, it is the dominant switching technique in digital telephone systems.

In the early seventies packet switching was introduced and established the foundation of data networks. Its main advantage over circuit switching lies in its wiser use of the bandwidth. In packet switching, data (regardless of its source) is divided into packets. Packets from different sources are transferred from one node to another until each packet arrives at its destination. No channels or circuits are reserved when the communication is idle. At the destination, packets are

assembled to reproduce the transmitted data, which is then conveyed to the receiver. Packet switching leads to a much better utilization of the bandwidth compared to circuit switching. Consequently it became the dominant switching technique in computer networks.

1.3.2 Integrated Service Digital Networks (ISDN) :

There is an increasing number of applications that require transmitting voice, video and data on the same communication channel. As a solution to such applications Integrated Service Digital Network (ISDN) was proposed [2].

The major deficiency with ISDN is its narrow bandwidth to carry different services. That is why it is sometimes called Narrow ISDN (N-ISDN). The fast development in the various digital communication systems led to applications that are very demanding in bandwidth. Around 1988 the telecommunications industry began to develop the concept of Broadband Integrated Service Digital Network (B-ISDN) [7]. The aim was to provide high-speed communications to the end users in an integrated way. B-ISDN standards were developed in a number of standards bodies around the world and were finalized by ITU-T. The initial ITU-T (that time it was CCITT) recommendations on B-ISDN were published in 1988 [8]. The ITU-T recommendations outlining the fundamental principles and initial specifications for B-ISDN were approved in 1990. The B-ISDN standards, protocol and layers were developed around the world.

During the development period, many *transferred modes* were studied and experimented for satisfying the objectives of B-ISDN. Transfer mode is usually used by ITU-T to describe the technique that is used in a network, covering the

aspects like transmission, multiplexing and switching [9]. At the beginning there were many candidates. Some of these candidates are circuit switching, multirate circuit switching, fast circuit switching, packet switching and fast packet switching [9]. Describing these techniques and analyzing their performance towards achieving B-ISDN is out of scope of this thesis. In section 2.4 of [9] a detailed study of their performance can be found. In summery, the first four techniques showed poor performance regarding time and/or the ability to tolerate integrated services. The fifth one (fast packet switching) showed good performance and was developed later to the asynchronous transfer mode (ATM).

Proposing ATM as the standard transport technique for use with B-ISDN was officially published by CCITT Recommendation *I.121*, "Broadband Aspects of ISDN" [7]. At the beginning ATM was strongly attached to B-ISDN to the level that researchers are using them interchangeably for most practical purposes [10]. Later ATM expanded and developed to become a useful technology in many different environments and applications that are not strictly B-ISDN [8].

Even though B-ISDN/ATM is basically packet switching oriented, it has many special features that make it much faster than all traditional packet switching techniques. B-ISDN can support very high data rates like 2.5 Gb/s or more [2]. In the following section will give a brief description of some ATM aspects like layers, cell format, switching and some other issues which are useful for general understanding of the subject and, at the same time, required for the remaining chapters.

1.3.3 ATM Networks:

ATM was independently proposed by Bellcore, the research arm of AT&T in the US, and several giant telecommunications companies in Europe in the first half of the eighties [7]. The basic idea of ATM was to carry the connection identifiers along with the data in any bucket instead of always identifying a connection by the bucket number. In the same time it keeps the size of the bucket small so that if any one bucket got dropped for any reason, not too much data would get lost, and in some cases could be easily recovered. This sounded very much like packet switching, so it was referred to as "Fast packet switching with short fixed length packets" [9]. Although some of the fundamental concepts of ATM can be traced back to packet switching systems, ATM differs significantly from traditional packet switching techniques. Its commercial availability marks the beginning of what promises to be a new revolution in both the data communications (related computer networks) and telecommunications (related to digital telephone networks) industries.

ATM cells are made small and of fixed size, which enables ATM to offer considerable flexibility because cells can be sent at any rate (within the limitations of the system) to suit the requirements of the users. ATM has a wide-bandwidth (low overhead) and small processing delay (simple and fixed cell format). ATM traffic is usually characterized by being asynchronous and also being connection-oriented. Characterizing ATM as asynchronous indicates that cells may occur at irregular times that are determined by the nature of applications rather than the framing structure of the transmission system. Characterizing ATM as connection-oriented spells out of the need to reserve some network resources in order to meet the application service requirements. It also indicates that each switch in the network keeps cell-routing tables [7], which tell the switches how to associate

incoming cells with the proper outgoing links (i.e., where to switch incoming cells). This is done using the VPI and VCI which will be explained later.

The almost universal acceptance (among supplies and developers) of ATM comes from the fact that ATM has the following features [7-13]:

- ATM can handle all the different kinds of communication traffic (voice. data, image, video, high-quality sound, multimedia, etc.) in an integrated way.

- ATM can be used in both the LAN and the WAN network environments and hence simplifies interworking between the two [10]. Thus it is effective in a much wider range of communications environments than any previous technology.

- ATM can handle different network data rates.

- ATM is a new technology designed to operate in the available technological environment.

With all the nice features mentioned above about ATM technology, it is still a compromise. Since it have some drawbacks like [7-13]:

- ATM does not handle voice as efficiently (or as cost effectively) as does the available networks.

- ATM does not handle video as easily as isochronous transfer dose (although it is a lot more efficient).

- ATM certainly dose not handle data as effectively or efficiently as a "packet transfer mode " or frame relay system.

- Cell loss problem during ATM cells switching can be a serious problem in some cases.

Nevertheless, ATM will handle all types of traffic perfectly adequately and in an integrated way. This means that, instead of having a group of many specialized kinds of equipment for different functions, we can have a single type of equipment and network which will do everything. ATM can do that in a performance which is very acceptable and has the ability for vast improvements in the future.

In 1992 industry forecasters were saying that ATM would begin an experimental phase in 1993, have early commercial products in perhaps 1997 and that the year 2000 would be the year of mass usage [8]. The first commercial products actually became available in 1994 and mass acceptance began in 1995. That happened even though many of the standards were scheduled for completion by end of 1996 [7].

1.3.3.1 ATM Layers:

ATM layers do not follow the international standard Open System Interconnection (OSI) model. However ATM made an extensive use of OSI concepts of layering and sub-layering [12]. There are three basic layers for ATM networks. Upper layers could be used if necessary, but they are not included in ATM/BISDN

standards. The three basic layers are physical layer, ATM layer and ATM Adaptation layer (AAL). Going to the detailed explanation about the functions of these layers and their sublayers is out of the scope of this thesis. Such details can be found in any of the sources [7-13]. we only review the main functions of these three layers.

The data reaches the AAL from the upper layer at any format. AAL job is to analyze that format and modify (or adapt) it to the ATM format [12]. Different applications may differ in some requirements such as synchronization and whether the bit transfer is constant or variable. There are five different AAL types to accommodate such different requirements. These layers are labeled AAL1, AAL2, AAL3, AAL4 and AAL5. There is a different ATM cell format for each of these adaptation layers as it will be explained shortly.

ATM layer is there simply to perform the special ATM functions like switching and multiplexing ATM cells [10]. The Physical layer perform its typical functions like: bit transfer/reception, bit synchronization, error control, and others. In Figure 1.4, these layers are illustrated with their general functions.

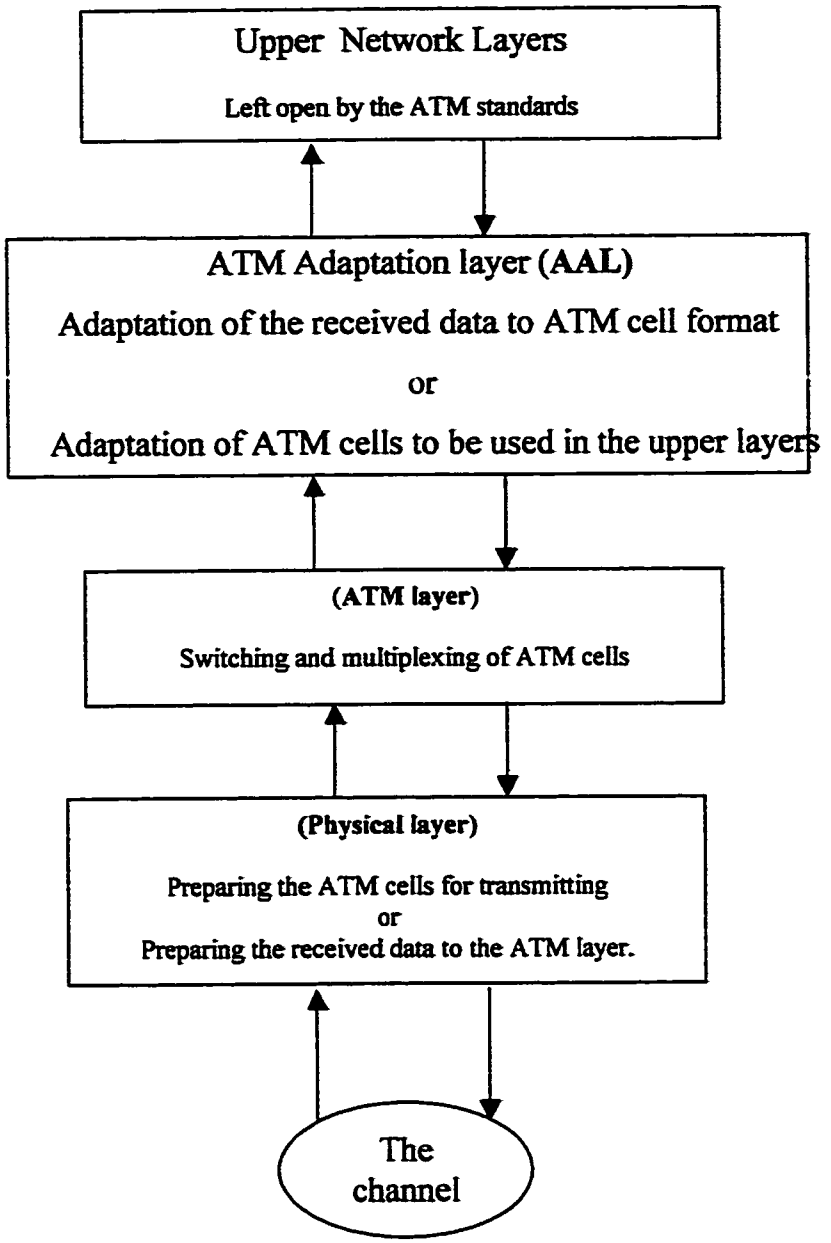


Figure 1. 4 ATM network Layers

1.3.3.2 ATM Cell Format:

ATM technology is based on small, fixed-size cells that permit sufficiently rapid switching for different types of data. The ATM cell consists of 53 bytes. Out of these there are 48 bytes that constitute the payload, and 5 bytes that constitute the header, as shown in Figure 1.5.

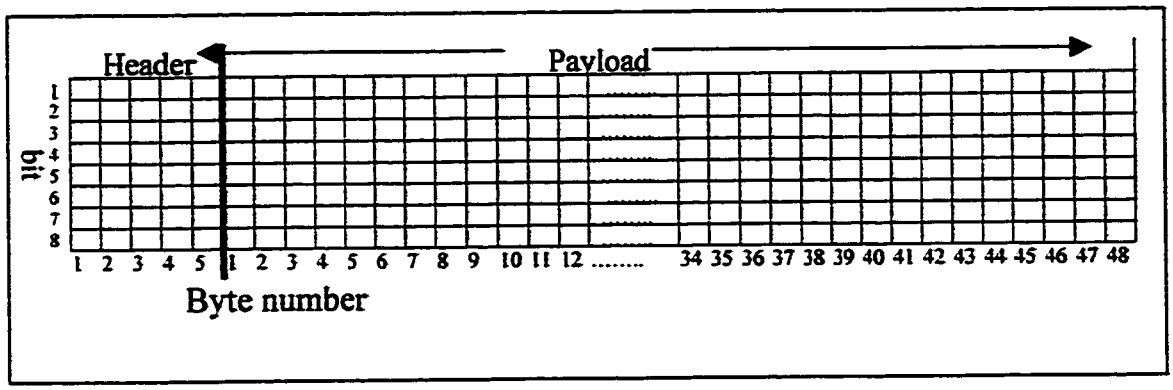
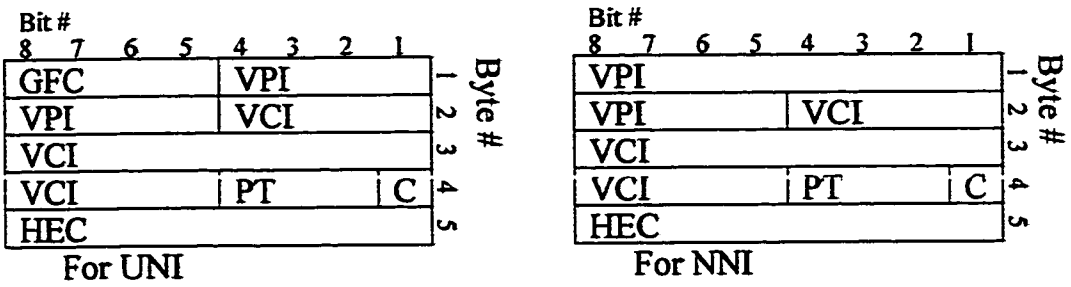


Figure 1. 5 General ATM Cell Format.

There are two types of headers, one is used for User-Network Interface (UNI) and the other one is used for Network Node Interface (NNI). They are similar except for the first byte. Both types are shown in Figure 1.6.



VPI : virtual path identifier.

VCI : virtual circuit identifier.

C : cell loss priority.

GFC : generic flow control.

PT : payload type.

HEC : header error check.

Figure 1.6 Structure of the Header of ATM cells for both the interfaces.

The ATM cell header illustrated in Figure 1.6 consists of five basic fields: VPI, VCI, PT, HEC and C. For the UNI only, the cell header also includes a generic flow control (GFC) field. This is a 4-bit field and it is part of the VPI inside the network. GFC provides a framework for flow control to the user-network traffic and does not control the traffic in the other direction (i.e., network-to-user traffic flow). If there is no interest to use GFC, it should be filled by zeroes [11].

The PT is used to indicate the payload type, i.e. whether it is network (maintenance and management) or user data. Also the PT is used for including the congestion notification bits.

C (or sometimes referred to as CLP) is a one bit field used for cell loss priority indication. C=0 implies high priority whereas C=1 implies low priority [12]. A clear example of that is in image coding where some of the cells are not essential but they are used for enhancing the image quality. If these cells are marked by 1 in the C field, they will be the first cells to discard during congestion hopping for minimum degradation of image quality.

HEC is a CRC-8 code parity to protect the valuable information in the header. This CRC is used in one of two modes: pure error detection or single error correction and multiple error detecting. When errors are detected (and can not be corrected), the cell is discarded.

VPI (8 or 12 bits) and VCI (16 bits) are the longest fields and the most important ones. They are used for cell switching between source and destination through the intermediate nodes. Because of the importance of these two fields, the header containing them must be protected be a CRC code to prevent miss-routing. ATM standards defined two types of ATM connections: Virtual Path Connections (VPC) and Virtual Channel Connections (VCC). VPC is identified by the VPI, while the VCC is identified by the VCI. A virtual path can be thought as a bundle of virtual channels carrying streams of cells, in order, from user to user. Or, in other words, the virtual path is like the trunk between two cities and the virtual channels are the established calls in that trunk passing through many intermediate switches [black]. Virtual channels are switched by VC switch and virtual paths by VP switches. During this process the values of incoming channels or paths identifiers must be converted into the new values for outgoing channels or paths.

1.3.3.3 Payload format:

There are different structures of the 48 payload fields corresponding to the different adaptation layers. Standardized payload structures for AAL1, AAL3/4 and AAL5 are shown in Figure 1.7. AAL2 structure is not yet standardized neither used in the applications [7-13].

We can see from Figure 1.7 that variable number of bytes (depending on AAL type) of the 48 payload bytes are used for control purposes. The 4-bit Serial Number (SN) is a very important field used in AAL1 and AAL3/4 to detect any lost cell or wrongly inserted cell in a series of cells [8]. The easiest way to detect lost cells is by checking this SN. This was used in [14-17]. The SN based detection works well for detecting random loss as will as bursts up to 16 cells. In [18] and [19], an attempt is made to utilize cell fields with the SN to extend the detection capability and increase the number of cells that can be placed in one row.

In AAL1 to protect the valuable information of the SN, another protection field is attached to it which is SNP.

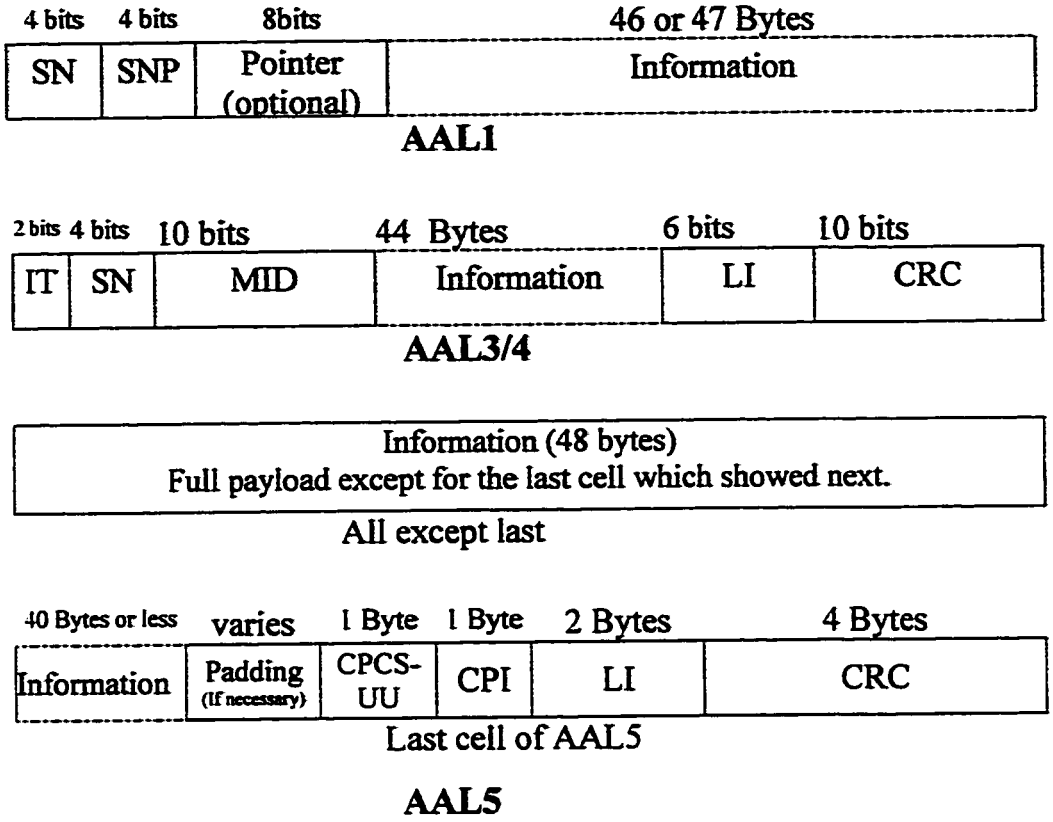
SNP consists of four bits also. Three bits of the SNP are the parity of the CRC-3 code for the four bits of SN, and the fourth bit is a parity check on the other three parity chicks. This is shown in Figure 1.8.

The MID is the Message or Multiplexing Identification used for identifying cells belonging to different data groups multiplexed in the same virtual connection. Information Type (IT) tells whether the cell is beginning, middle or bottom of the message. The fourth option of the two-bit IT goes to indicate that this cell carries a single cell message (single segment).

The length indicator, (LI) gives the number of information bytes in the cell for AAL3/4 or in the whole message for AAL5. In AAL3/4, the value of LI can be maximally 44 bytes. In AAL5, the message is segmented to cells of 48 bytes (full load) without any control fields. In the last cell of the message some control fields are attached to control the whole massage. The LI in the last cell consists of two bytes giving the possibility of counting up to $2^{16} = 655,36$ information bytes.

CRC is a cyclic redundancy check parity to detect any error in the payload. It is CRC-10 for AAL3/4 and CRC-32 for AAL5. It is stronger in AAL5 because it checks the whole message while in AAL3/4 it checks one cell only.

The two remaining fields in AAL5 which are described above are CPI and CPCS-UU. They are not fully defined by the standards [8] but they are used to identify the common part and the user part of the data.



Where:

SN	Serial Number.	SNP	Serial Number protection.
LI	Length Indicator.	IT	Information Type.
CRC	Cyclic Redundancy Check.	MID	Message Id.
CPI	Common Part Id.		

CPCS-UU Common Part Convergence Sublayer-(User to User) indication.

Figure 1.7 the format of payload part of ATM cells for different AAL types.

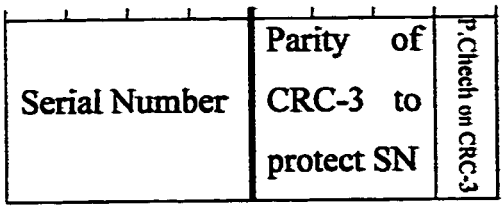


Figure 1.8 The serial number and its protection in AAL1

1.3.3.4 Cell loss in ATM networks:

ATM networks suffer from lost and discarded cells during transmission. Buffer overflow due to congestion is the main source of cell loss in ATM networks. Cell discarding is mainly due the error detection in the header. The header contains the valuable information of VPI and VCI that are responsible for routing. This means that if an error occurs in the header it will most likely yield miss-routing of the cell. Miss-routed cells are useless an hence they are discarded. There are other minor sources of cell loss (discarding) like when a cell is dropped because it does not satisfy the required QoS (Quality of Service) [10]. Cell loss seriously degrades transmission quality. In this thesis we will analyze a simple code the (PCPC) for the purpose of lost cells recovery. When the code is applied for recovering lost cells in ATM networks, we will assume that cell loss is only due to non-correctable error in the header.

1.4 Literature Survey:

Different recommendations are given for dealing with the problems encountered in transporting ATM cells in [20]. The recommendations are related to three major approaches. The first one is related to cell format and layer protocols. The second approach is about using error correcting and/or error detecting codes (FEC /ARQ). The last set of recommendations is related to some special data processing like interleaving.

Some researchers studied the use of ARQ alone as error control in ATM networks. An example can be seen in [21], where two ARQ schemes are used for error control. The first is Go-Back-N alone and the second is adaptive Selective-Repeat with Go Back-N (SR+GBN) ARQ. To achieve better results a modified format of the ATM cell was used and an acknowledgment cell format was designed. For Bit Error Rate (BER) less than 10^{-6} the throughput was found to be very high for both schemes.

In [22] three error control techniques are investigated for ATM cells transmitted over wireless communication. The techniques are: Pure Selective Repeat ARQ (SR-ARQ), Type-I Hybrid SR-ARQ/FEC, and Type-II Hybrid SR-ARQ/FEC. The authors used BCH codes and RS codes for error correction. They further suggest that according to QoS required, network connections can switch between the three strategies. The schemes in [22] work well at low BER. For high BER, all the control strategies give poor throughput.

In [23], in order to improve the performance of wireless ATM communication two error correcting codes are applied. The first FEC code is a fixed rate code applied on the header, while the second FEC code is a variable rate code applied on the payload. The rate of the code is adaptively changing depending on the channel condition and QoS required. QoS information is included in the header. Coding the header with (28,16) double error-correcting code gave a huge gain over the uncoded header. This gives higher protection for the header including QoS information, which is needed to give better protection for the payload. The codes of different rates, used to protect the payload, are constructed by shortening the same original code. This permits using the same encoder and decoder for all the different code rates. The authors concluded that

this method results in significant decrease in the cell loss rate at the price of a minimal increase in bandwidth.

Concatenated codes are used to improve the reliability in ATM networks in [24]. The inner code used is a convolutional code of rate=0.5 and memory of order 6. The outer code is an (n, 53) RS Code, where n varies between 53 and 63 bytes. Interleaving is used to randomize the errors. The result shows clear improvement compared to the uncoded communication. If ARQ is also to be used, the authors suggest to use a convolutional code with higher rate RS-code.

In [25] the effect of FEC on ATM transmission is analytically studied. The code used is Reed-Solomon Erasure (RSE) code. The correlation between cells is used in the analysis to investigate how this correlation can affect the cell loss rate. Also, traffic characteristics such as peek arrival rate, burstiness and burst length are included in the analysis. The impact of buffer size is addressed. This work results show that FEC can be very effective in reducing burst cell loss (whole block) if the correlation between cells is not strong.

Another trial to improve the reliability of ATM communication can be found in [26]. It is shown by simulation that transmission reliability over wireless channels, which are naturally bursty error channels, can be improved by redesigning the transmitted cells. ATM cells go through a procedure of interleaving, coding, fragmentation and puncturing to give the final transmitted cells.

Two different FEC methods are studied when used in ATM cells transmission via satellite links in [27]. The first is a concatenated code used for the ATM cell header. The outer code is the standard cyclic code (40,32), while the

inner code is convolutional code. Interleaving is used between the inner and outer codes. In the second method, the header is coded using RS(14,8). The coded header with payload are again coded using RS(255,215).

Other recent works in the field of error control in ATM networks can be found in [28-30]. However, in some of these systems [20-30], the ATM cell format is modified before being transmitted over the network. Modification is due many operations like: coding, interleaving, puncturing and others. This modification is disliked due to introducing some sort inhomogeneous frame of work, requiring extra control and the possible delay introduced. Also, the complexity of coding/decoding in most of the above systems may be considered a real practical implementation problem.

A more practical scheme would be that which preserves the cell format and keeps the encoding and decoding schemes simple. A simple FEC technique is proposed in [15], which is a single parity check code. Cells from the same path are grouped in a matrix form. An M^{th} row is added as a single parity check for cell recovery. This method can recover one lost cell per column. It is also capable of recovering a burst of N lost cells, where N is the row size.

In [16,17] a detailed analysis of a hybrid ARQ/FEC coding scheme is given. The FEC used is the simple technique used in [15] (only a single parity check). Go-Back-N ARQ is utilized to improve system performance. The system performs the error control on the VC instead the VP as it was in [15]. The paper analyzes the probability of unrecovered lost cells, throughput and reliability of the proposed system. The analysis includes many network parameters in evaluation the performance of the system.

In [14] a major improvement of the simple FEC technique in [15-19] is made by extending the parity check for lost cells recovery to be two dimensional. In other words, a two dimensional parity check recovery was applied trying to restore any lost cells. That was done simply by placing one column at the right of the matrix as a row-wise parity check and one row at the bottom of the matrix as a column-wise parity check. By this method more lost cells patterns can be corrected. The analysis of cell loss rate in [14] shows a clear improvement over the one-dimensional case for the same amount of redundancy.

1.5 Proposed Work:

In this work we analyze a forward error correction (FEC) and hybrid ARQ/FEC schemes for operation over an ATM link. The FEC scheme adopted here is the parity check product code (PCPC) while the ARQ scheme is selective-repeat ARQ (SR-ARQ).

The work in [14] provides a primitive attempt to evaluate the capability of PCPC to recover lost cells. The authors derived a bound on post decoding cell loss rate that turns out to be very loose for moderate and large cell loss rate. The complexity of the problem lies in identifying and counting the unrecoverable lost cell patterns.

This work introduces a novel algorithm for structural analysis for the lost cell patterns in PCPC. The algorithm is based in constructing a small set of lost cell patterns called *basic patterns,* from which all lost cell patterns can be

generated. This approach leeds to a systematic way of identifying recoverable patterns. Although emphasis is made on PCPC, the approach is general and can be applied to any class of product codes.

The results of the novel algorithm were used to derive three upper bounds on post decoding cell loss rate of PCPC. Each bound has a degree of tightness in proportion to the degree of complexity . In fact, every bound is obtained by tightening of the previous bound. From that result Bound III can be considered the ultimate result of this part of the work, with Bound I and Bound II being intermediate steps of derivation. The system was simulated in order to evaluate the tightness of the bound.

The work up to this point assumes no special format for the cell in the product code. It could be a bit, a symbol or a packet of bits or symbols. In the last part of this work, the PCPC is applied to ATM cells. Both pure PCPC and hybrid ARQ/PCPC are considered.

For the pure PCPC , the performance of the code in terms of post – decoding cell loss rate versus channel bit error rate (BER) is evaluated. The code is compared with two other codes, the one–dimensional parity check code in [15] and the RS code.

For the hybrid ARQ/PCPC scheme, an SR-ARQ is implemented on the top of PCPC to improve cell loss recovery. Two retransmission strategies are investigated, the full matrix retransmission and only lost cell retransmission. The hybrid scheme is evaluated in terms of its throughput efficiency and compared with pure ARQ.

The proposed work is organized in three chapters. The novel algorithm for structural analysis of PCPC is explained in Chapter 2, the bounds on post-decoding cell loss rate are derived in Chapter 3 and the application of PCPC, both pure and hybrid, to ATM link is studied in Chapter 4. We conclude with chapter 5, where we summarize the findings of this work an the directions for future work.

CHAPTER 2

A NOVEL STRUCTURAL ANALYSIS OF LOST CELL PATTERNS IN PRODUCT CODES

2.1 Introduction:

The code under investigation is the parity check product code (PCPC). The code is constructed by arranging the cells in a matrix of size $(M\text{-}1)\times(N\text{-}1)$, then adding a parity cell to each column and a parity cell to each row, resulting in $M\times N$ matrix. It is assumed that the $M\times N$ matrix is received with some of the cells lost (erased).

The decoder will perform alternating row-wise and column-wise decoding to recover the lost cells. When a single cell is lost in a row or column, it can be

recovered by a simple parity checking (mode-2 addition). If more than one cell is found lost in a row (column), that row (column) is skipped. Decoding will be performed in many rounds until all cells are recovered or no further recovery is possible.

Using this decoding scheme, a pattern is unrecoverable pattern if and only if it contains a sub-pattern such that every occupied row and every occupied column in that sub-pattern has at least two lost cells. This condition can be stated as follows. Let (i ,j) denotes the (row , column) location of a lost cell. A sub-pattern is unrecoverable if:

> For every lost cell in location (i ,j), the cells in locations (k, j) and (i , l) are also lost; where k can take an value from 1 to M but $k \neq i$ and l can take an value from 1 to N but $l \neq j$.

Clearly, the PCPC can recover all single-, double- and triple- lost cells patterns regardless of how they are distributed in the matrix. Patterns consisting of more than three lost cells may or may not be recoverable depending on the way they are distributed in the matrix. Examples of recoverable and unrecoverable patterns are shown in Figures 2.1 and 2.2 respectively.

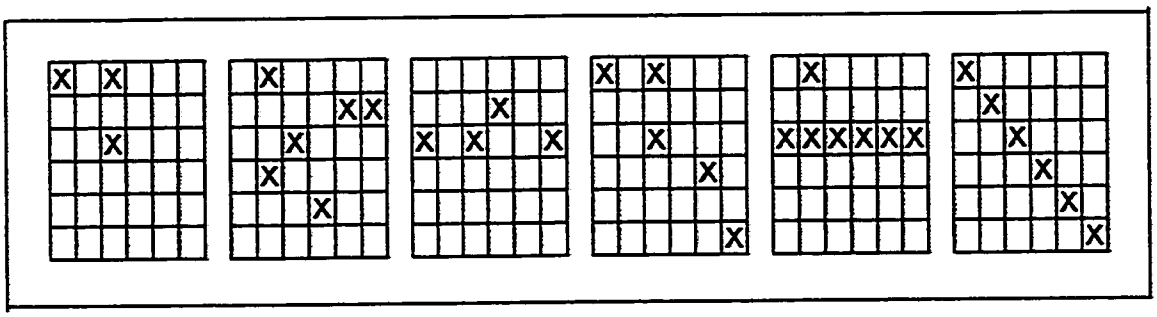


Figure 2. 1 Examples of recoverable patterns.

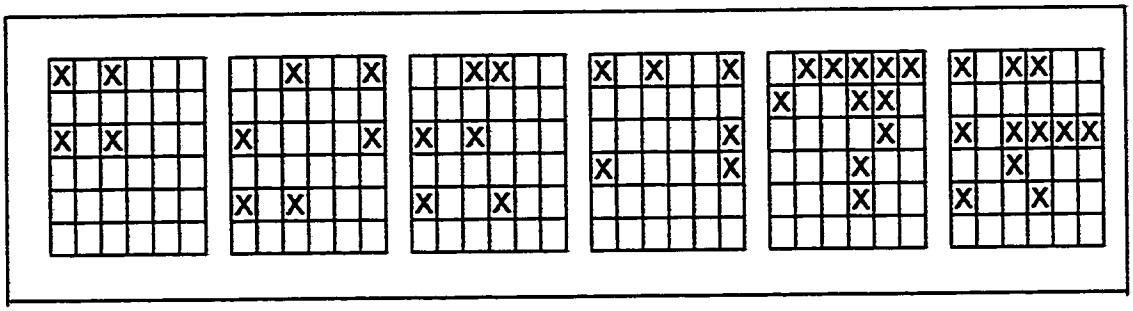


Figure 2.2: Examples of unrecoverable patterns.

The code is evaluated in terms of post decoding Cell Loss Rate (CLR). Let i be the number of lost cells and U_i be the number of unrecoverable patterns consisting of i lost cells. Obviously, $U_1=U_2=U_3=0$. Then the post–decoding CLR P_d will be [14]:

$$P_d=\frac{1}{MN}\sum_{i=4}^{MN} iU_iP_l^i(1-P_l)^{MN-i} \qquad (2.1)$$

where P_l is the pre-decoding CLR.

To evaluate Equation (2.1), we need to evaluate U_i for $i\geq 4$. The authors in [14] evaluated U_4 and U_5. They assumed that all patterns of six or more cells are unrecoverable which yields a very loose bound. In the next chapter we will derive two bounds for U_i for $i\geq 6$. Finding these bounds depends on the structure of lost cell patterns in a matrix which is explored in this chapter.

The remaining sections of this chapter are organized as follows. Motivations for the structural analysis and some preliminaries on recoverable patterns are given in the following section. The new concept of *basic patterns* and the way of generating all the remaining possible patterns from them are introduced in the following two Sections (2.3 and 2.4). In Sections 2.5 and 2.6 the basic patterns will be classified into recoverable basic patterns and unrecoverable basic patterns .

2.2 Motivation for the structural analysis for lost cells

Any number of lost cells i can be distributed in a matrix of size $M \times N$ in many different ways. We will be calling each of these distributions a ***pattern***. The number of lost cells in a pattern is called the ***pattern size***.

Recall that the total number of possible patterns $= \binom{MN}{i}$, where the term $\binom{X}{Y}$ is the binomial coefficient.

As mentioned before, there are no unrecoverable patterns of size one, two or three. Larger size patterns are considered in the following paragraphs.

Unrecoverable patterns of size four U_4:

It is easy to see that the only unrecoverable patterns of size four are those where the lost cells form the corners of a rectangle, as shown in Figure 2.3.

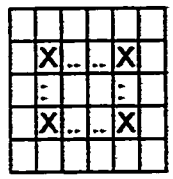


Figure 2.3: The general shape of unrecoverable pattern of size 4.

For a matrix of M rows and N columns, [14] showed that the number of unrecoverable patterns of size four, U_4, is:

$$U_4=\binom{M}{2}\times\binom{N}{2} \qquad (2.2).$$

Unrecoverable patterns of size five U_5:

The only unrecoverable patterns of size five are those which are constructed from size-4 unrecoverable patterns with the fifth cell located anywhere else in the matrix [14]. Therefore the number of unrecoverable patterns consisting of five cells, U_5, is:

$$U_5=\binom{M}{2}\times\binom{N}{2}\times(M\times N-4) \qquad (2.3).$$

Unrecoverable patterns of size ≥ 6:

It was assumed in [14] that the lost cells patterns of size 6 or more are all unrecoverable; that is:

$$U_i=\binom{M\times N}{i} \quad for\ i=6,7\ldots\ldots,M\times N. \qquad (2.4)$$

Clearly, this leads to a very loose bound on the post-decoding CLR. The authors were driven to this assumption because it was not feasible how to identify and enumerate the unrecoverable patterns for larger number of lost cells. Clearly there is a need for a systematic approach to identify the unrecoverable patterns because exhaustive search is computationally prohibitive. In this chapter, we are proposing a novel approach for classifying the structures of lost cell patterns in a matrix. The basic idea in this approach is to identify some special distributions of cells (patterns) called *basic patterns* which can generate all the possible patterns. This approach will make it possible to identify and enumerate the unrecoverable patterns for large number of lost cells. The results of this chapter will be used in Chapter 3 to develop tighter bounds on post decoding CLR.

2.3 Basic Patterns

The number of different patterns of lost cells depends on the number of lost cells and the matrix size. Out of all these different patterns we will introduce a subset and call it ***basic patterns***

The basic patterns for a specific number of cells are those patterns that satisfy the following two criteria:

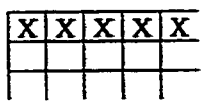
1. **The cells in each column are consecutive, starting from the first row,**
2. **The cells in each row are consecutive, starting from the first column.**

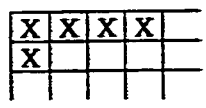
Any *basic pattern* satisfying the above criteria is characterized by the number of occupied columns and the number of cells in each of these columns. Any two distinct basic patterns differ from each other by the number of occupied columns and/or the number of cells in these columns.

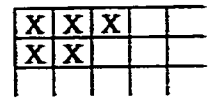
Examples of basic patterns and non-basic patterns are given in the following examples.

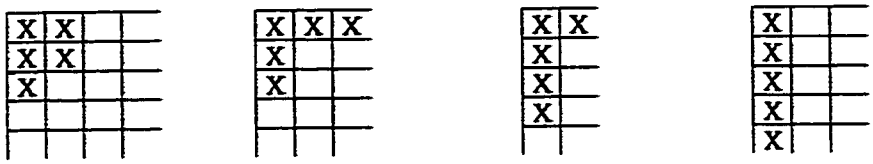
Example 2.1

For five lost cells, there are seven basic patterns:



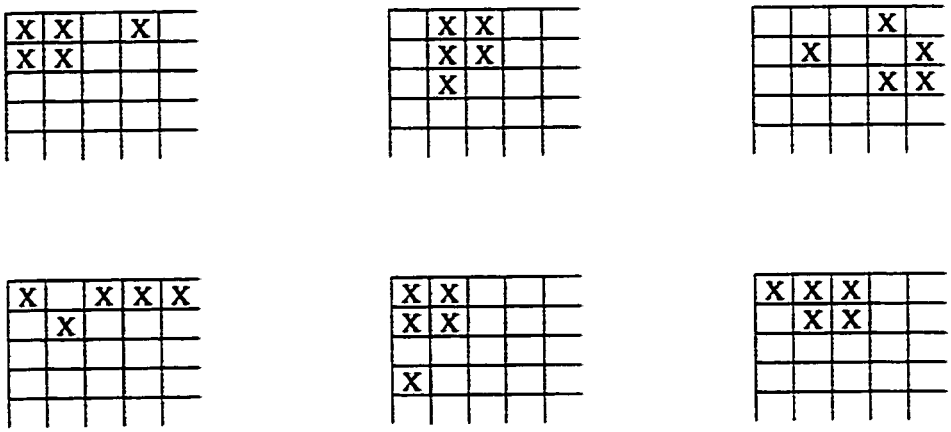






Example 2.2

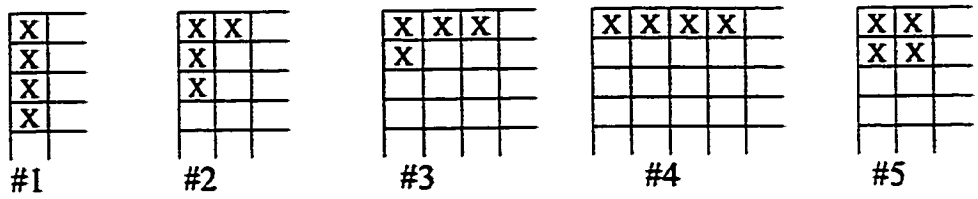
For five lost cells, the following patterns are ***not*** basic patterns:



The seven basic patterns in Example 2.1 are the only basic patterns of size five for any matrix of size $\geq 5\times5$. In general, for i lost cells the same set of basic patterns can be formed, regardless of the matrix size, provided that $i \leq M,N$. If any of the matrix dimensions is smaller than i, then the number of basic patterns will decrease. The above concept is illustrated with the following example.

Example 2.3

The set of basic patterns for 4 cells has the following five basic patterns. These basic patterns are the same for any matrix of $M,N \geq 4$.



However, for any matrix smaller than 4×4, the number of basic patterns for 4 lost cells will be less than five. The following table shows the 4-cells basic patterns for different sizes of matrixes.

Matrix size	Basic patterns that can fit
$M \geq 4$ and $N \geq 4$	All the five basic patterns.
M×3 $(M \geq 4)$	#1, #2, #3 ,#5
3×N $(N \geq 4)$	#2, #3 ,#4, #5
3×3	#2, #3 ,#5
3×2	#2, #5
2×3	#3, #5
2×2	#5

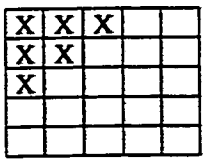
2.4 Generated Patterns

The patterns produced from any basic pattern (including the basic pattern itself) are called ***generated patterns***. The generated patterns are generated from a basic pattern by allowing the occupied columns and their cells to move everywhere possible in the matrix provided that the number of the occupied columns and the number of cells in each column are preserved.

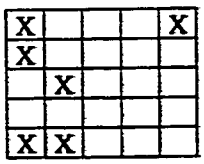
Therefore, the relation between a basic pattern and its generated patterns is that they all have the same number of occupied columns with the same number of cells in each column. They differ in the particular columns they occupy and/or the distribution of the cells within these columns.This is illustrated in the following example.

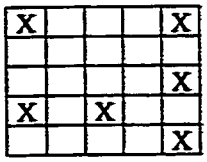
Example 2. 4

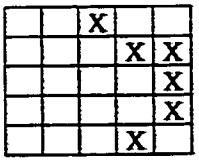
Consider the following basic pattern consisting of six sells:

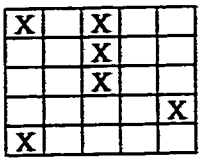


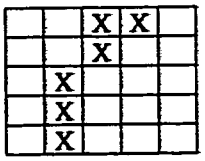
Some of the patterns generated from the above basic pattern are:

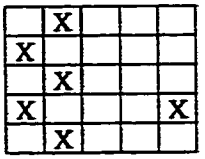












Note that each one of these generated patterns has three occupied columns with 1,2 and 3 cells.

Now we state the following important result.

Result 2.1:

For any number of cells, the generated patterns from all basic patterns are distinct and exhaustive i.e. they span all the $\binom{MN}{i}$ possible patterns of i cells in the matrix.

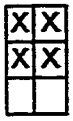
The fact that the generated patterns are distinct is obvious. It follows from the structure of the basic pattern and the method of producing the generated patterns. Basic patterns are distinct with regards to the number of columns and the number of cells in each column, and this distinction is preserved between their generated patterns. Hence all the generated patterns are distinct.

The generated patterns from all the basic patterns are exhaustive. This is so because the basic patterns exhaust all the possible number of occupied columns and all the possible distribution of cells in these columns.

The following examples demonstrate Result 2.1.

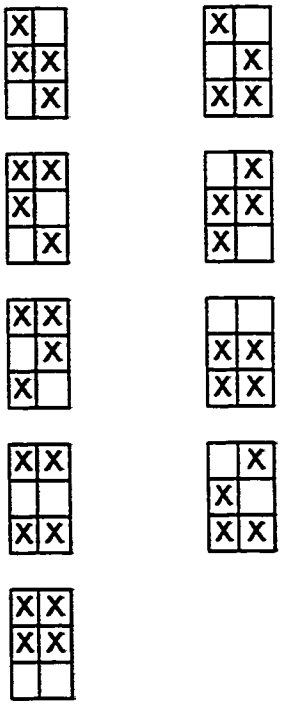
Example 2. 5

4 cells in a 3×2 matrix has only two basic patterns:

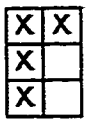


Basic Pattern 1.

Generated patterns:

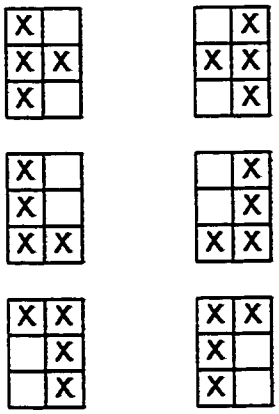


Total of 9 generated patterns:



Basic Pattern 2.

Generated patterns:

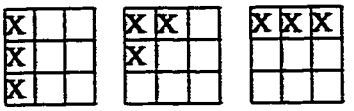


Total of 6 generated patterns:

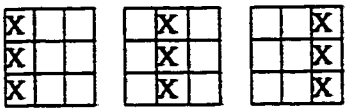
The Total Number of patterns = 9+6 =15, which equal to $\binom{6}{4}$.

Example 2. 6

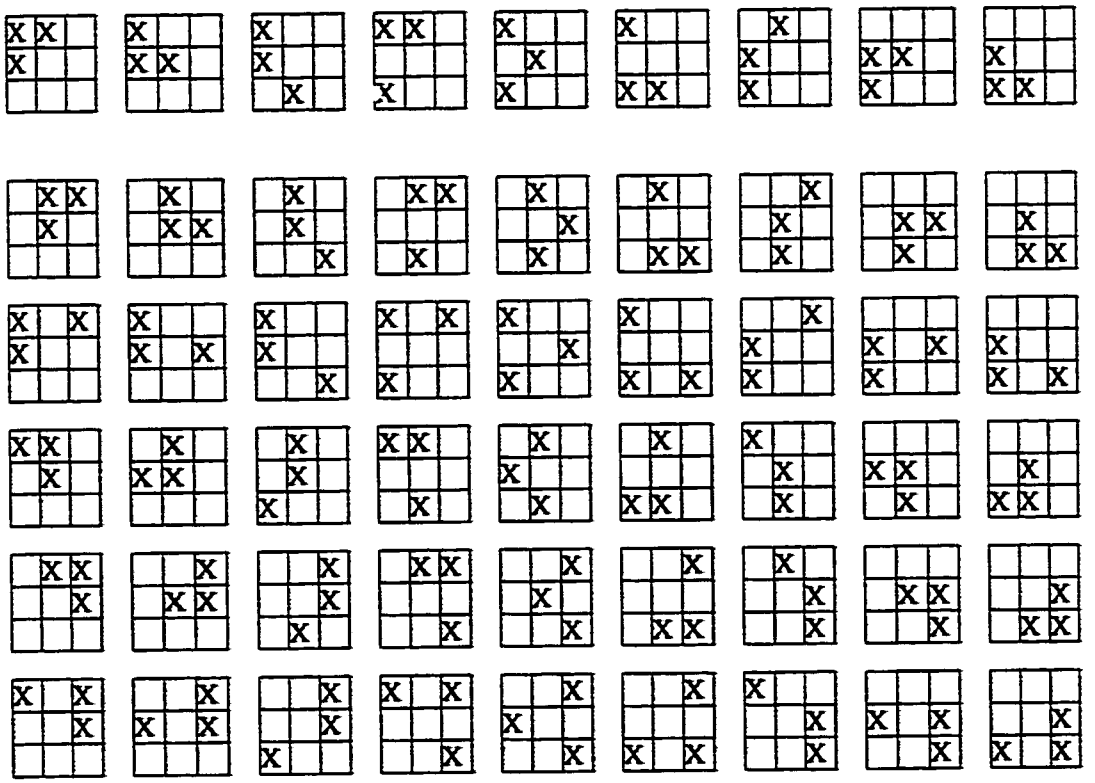
The following are the three basic patterns for 3 cells:



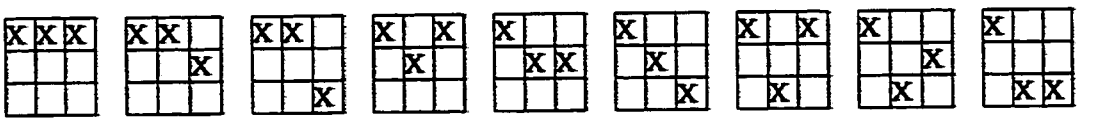
The first basic pattern can generate only three patterns:

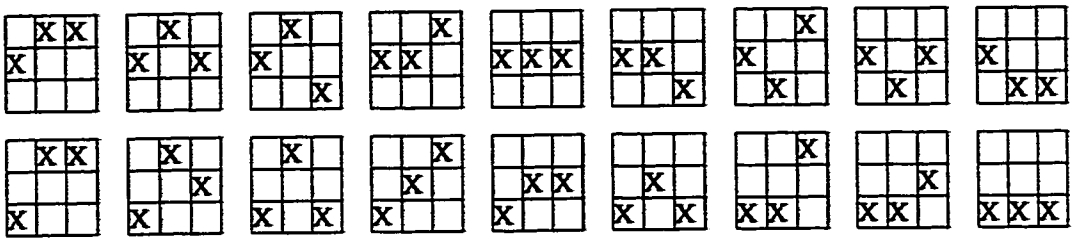


The second generated pattern can generate the following 54 pattern:



The third basic pattern can generate the following 27 patterns:





The total generated patterns = 3+27+54=84, which is the same as $\binom{9}{3}$.

In the above examples, the generated patterns from a basic pattern were obtained by exhaustive search. The number of generated patterns (G) from a basic pattern depends on the matrix size, number of cells in the basic pattern and the shape of the basic pattern. The number of patterns generated from a basic one is stated in the following result.

Result 2.2:

For any basic pattern p the number of generated patterns G_p is given by :

$$G_p = T_{p,1} \times T_{p,2} \times T_{p,3} \qquad (2.5)$$

where:

1. $T_{p,1}$ is the number of ways the cells can be distributed over the columns they occupy.
2. $T_{p,2}$ is the number of ways of distributing the occupied columns over the matrix columns without ordering (i.e. (1,2) and (2,1) are counted as one event).

3. $T_{p,3}$ is the number of arrangements of the occupied columns within themselves. If all the occupied columns have the same size, $T_{p,3}=1$.

It should be clear from result 2.1 that summation of G_p's for all the basic patterns is $\sum_{p=1}^{b} G_p = \binom{MN}{i}$, where b is the total number of basic patterns.

$T_{p,1}$, $T_{p,2}$ and $T_{p,3}$ can be evaluated using the following formulas:

$T_{p,1} = \binom{M}{C_1} * \binom{M}{C_2} * \ldots\ldots\ldots \binom{M}{C_n}$, which can be expressed in the form:

$$T_{p,1} = \prod_{i=1}^{n} \binom{M}{C_i} \tag{2.6}$$

Where: M *is the number of matrix rows (full column size).*

C_i *is the number of cells in the* i^{th} *column.*

n *is the number of occupied columns.*

$$Tp,_2 = \binom{N}{n} \tag{2.7}$$

where: *N is the number of matrix columns (full row size).*

$T_{p,3}$ depends on how many <u>different sizes</u> of occupied columns the basic pattern has. Let the number of different sizes be z and let z_j is the number of columns of size j. Then:

$$T_{p,3} = \prod_{j=1}^{z} \binom{n - x_{j-1}}{z_j} \quad , \tag{2.8}$$

where: $x_j = x_{j-1} + z_{j-1}$ and $x_0 = 0$.

For example when all occupied columns have the same size:

$z=1$ $z_1=n$ and consequently $T_{p,3} = \binom{1-0}{1} = 1$.

And when $z=3$ we obtain:

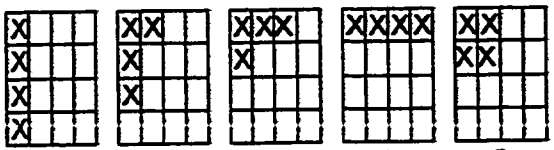
$$T_{p,3} = \binom{n}{z_1}\binom{n-z_1}{z_2}\binom{n-z_1-z_2}{z_3}$$

Note that the last term is always =1.

Let us now give an example to demonstrate the above equations. This example also serves as a higher-dimensional example demonstrating that the generated patterns of all the basic patterns are all the possible patterns for a given number of lost cells (Result 2.1).

Example 2. 7

Consider four lost cells in 4×4 matrix. There will be five basic patterns:



For basic pattern #1:

$$T_{1,1}=\binom{4}{4}=1, \qquad T_{1,2}=\binom{4}{1}=4, \qquad T_{1,3}=\binom{1}{1}=1,$$

$G_1=1*4*1=4.$

For basic pattern #2:

$$T_{2,1}=\binom{4}{3}\binom{4}{1}=16, \qquad T_{2,2}=\binom{4}{2}=6, \qquad T_{2,3}=\binom{2}{1}\binom{1}{1}=2,$$

$G_2=16*6*2=192.$

For basic pattern #3:

$$T_{3,1}=\binom{4}{2}\binom{4}{1}\binom{4}{1}=96, \qquad T_{3,2}=\binom{4}{3}=4, \qquad T_{3,3}=\binom{3}{1}\binom{1}{1}=3,$$

$G_3=96*4*3=1152.$

For basic pattern #4:

$$T_{4,1}=\binom{4}{1}\binom{4}{1}\binom{4}{1}\binom{4}{1}=256, \quad T_{4,2}=\binom{4}{4}=1, \quad T_{4,3}=\binom{1}{1}=1,$$

$G_4=256*1*1=256.$

For basic pattern #5:

$$T_{5,1}=\binom{4}{2}\binom{4}{2}=36, \quad T_{5,2}=\binom{4}{2}=6, \quad T_{5,3}=\binom{1}{1}=1,$$

$G_5=36*6*1=216.$

The total number of generated patterns = 4+192+1152+256+216= 1820, which is exactly equal to $\binom{16}{4}$.

The work so far proves that from very few patterns, the basic patterns, all possible patterns of i lost cells in a matrix can be generated. It is interesting to note that:

1. The number of basic patterns is extremely small compared to the number of possible patterns for a given i and $M\times N$.
2. The number of basic patterns increases only slightly with increasing the number of lost cells i for a given $M\times N$.
3. The number of possible patterns increases dramatically with increasing the matrix size $M\times N$ while the number of basic patterns is independent of the matrix size for $M,N \geq i$.

The first two observations are illustrated in the following example.

Example 2. 8

For a matrix of size 16 ×16, the following table shows the number of basic patterns and the number of total generated patterns (all possible patterns) for number of lost cells ranging from four to eighteen:

Number of lost cells	Number of Basic patterns	Number of Generated Patterns (G)
4	5	1.75E+08
5	7	8.81E+09
6	11	3.69E+11
7	15	1.32E+13
8	22	4.10E+14
9	30	1.13E+16
10	42	2.79E+17
11	56	6.24E+18
12	77	1.27E+20
13	101	2.39E+21
14	135	4.15E+22
15	175	6.69E+23
16	229	1.01E+25
17	293	1.42E+26
18	381	1.89E+27

Table 2. 1: Number of total generated patterns compared to number of basic patterns for different number of lost cells in a 16×16 matrix.

Table 2.1 tells us that, for example, any one of the patterns consisting of 6 cells in 16×16 matrix (about 369 billions) has the same structure of one of the 6 basic patterns. By same structure we mean the same number of occupied columns and the same number of cells in each of them. This observation will pave the road for classifying the possible patterns that *i* lost cells into recoverable and unrecoverable patterns.

2.5 Recoverable Basic Patterns

The basic patterns can be classified to recoverable basic patterns (RBP) or unrecoverable basic patterns (UBP). An important feature of any RBP is that its cells are located in the first column and first row of the matrix only. In other words no cells are found in the inner part of the matrix. An RBP has its cells arranged as in Figure 2.4

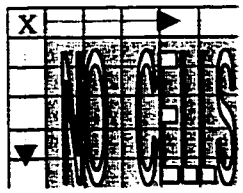


Figure 2.4: General shape of recoverable basic patterns

Where the arrows in Figure 2.4 indicates the possible places for the cells.

We should keep in mind that, since this is a basic pattern, the upper left cell should be occupied and all the cells in the column or the row should be adjacent. It is easy to see that these basic patterns are recoverable and that at most two decoding rounds (column-row or row column) are sufficient to recover all lost cells.

A careful study of the recoverable basic patterns leads to the following important result.

Result 2.3:

For any number of cells i in an M×N matrix, the number of RBPs (q_i) is:

$$\begin{aligned} q_i &= i && \text{if} && i \le \min(M,N) \\ q_i &= \min(M,N) && \text{if} && \min(M,N) < i \le \max(M,N) \\ q_i &= M+N-i && \text{if} && \max(M,N) < i \le M+N-1 \\ q_i &= 0 && \text{if} && i \ge M+N \end{aligned} \qquad (2.9)$$

These four intervals are depicted in the following sketch.

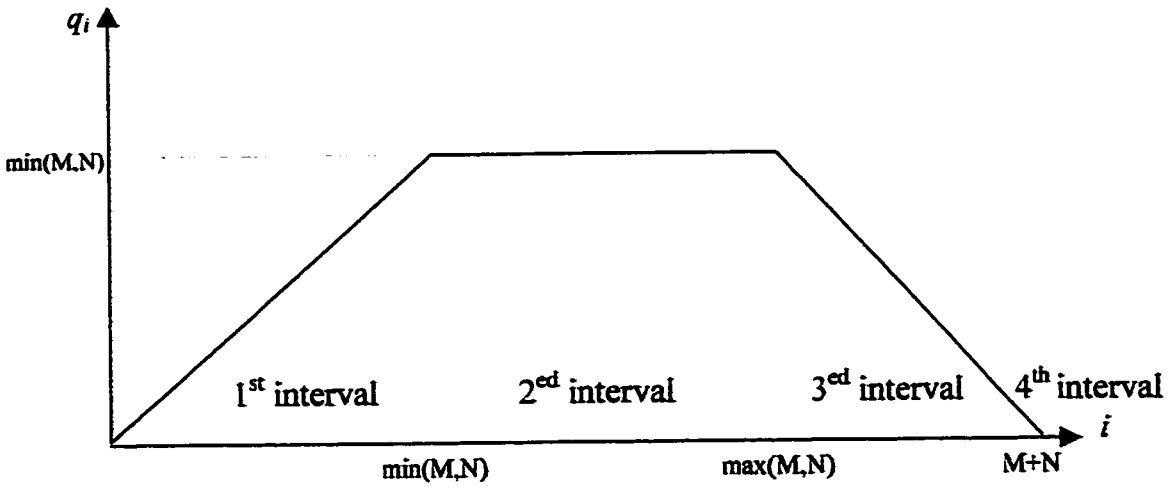
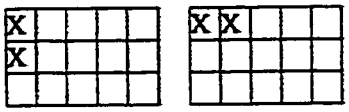


Figure 2.5 : relation between the number of cells and q_i in M×N matrix.

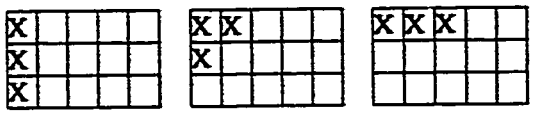
Example 2.9

For a 3×5 matrix, let's find q_i for different i's and show the shapes of these q_i RBPs:

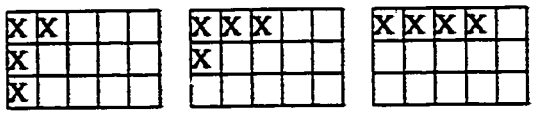
i=2 is in the first interval $\Rightarrow$ $q_2 = 2$



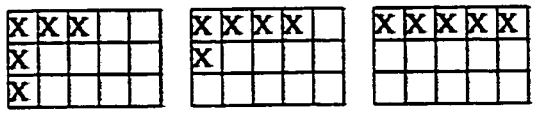
i=3 is in the first interval $\Rightarrow$ $q_3 = 3$,



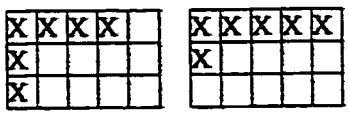
i=4 is in the second interval $\Rightarrow$ $q_4 = 3$,



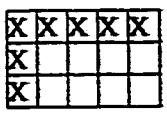
i=5 is in the second interval $\Rightarrow$ $q_5 = 3$,



i=6 is in the third interval $\Rightarrow$ $q_6 = 2$,



i=7 is in the third interval $\Rightarrow$ $q_7 = 2$,



i>7 is in the fourth interval $\Rightarrow$ $q_i = 0$.

Example 2.10

For a matrix of size 16x16, the following table shows the number of recoverable basic patterns for each number of cells:

Number of lost cells (i)	Number of Recoverable Basic patterns
1 to16	Same as i
17	15
18	14
19	13
20	12
21	11
22	10
23	9
24	8
25	7
26	6
27	5
28	4
29	3
30	2
31	1
32 to 265	0

Table 2. 2 : Number of RPBs a16×16 matrix for different number of cells.

The algorithm for constructing all the RBPs is given in Appendix A.

2.6 Unrecoverable Basic Patterns (UBP's)

After excluding the RBP's, all the remaining basic patterns will be called unrecoverable basic patterns (UBP's). The previous section shows that it is relatively simple and straightforward to enumerate and construct all the RBPs. Unfortunately, this is not the case for UBPs. There are many different general shapes of the UBPs, compared to the single general shape of RBPs which makes the procedure of constructing them more involved. In fact, the number of the UBP's increases faster than RBP's when the number of cells is increased. Moreover, there is no simple formula to find the number UBP's as compared to RBP's.

2.6.1 Generating UBPs: Algorithmic Approach:

For the basic pattern to be unrecoverable, all the UBP's contain the upper left 2x2 cells as a part of it. This is shown in Figure 2.6.

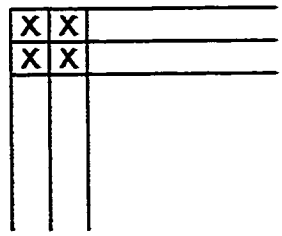
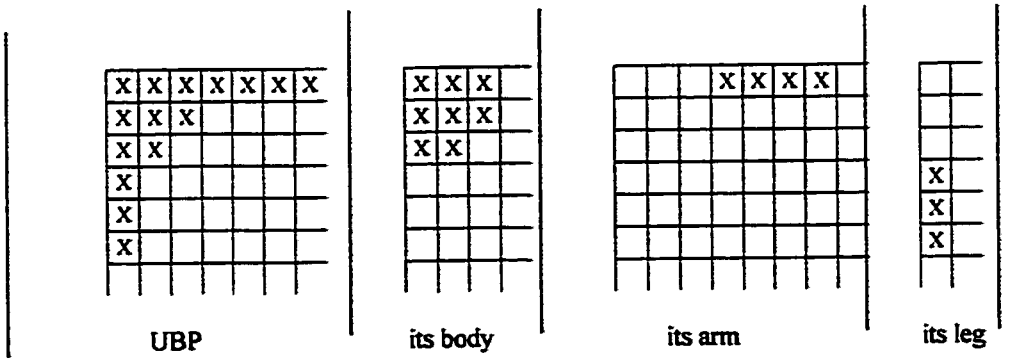


Figure 2.6 : the common part of all unrecoverable basic patterns.

Let's partition any UBP to a ***body*** and two branches called *arm* and *leg*. The arm is formed of the cells of the single-cell columns, whereas the leg is formed of the cells of the single-cell rows. The remaining cells of the UBP form the body. It is worth noting that in RBP's all columns and rows are single-cell. Therefore an RBP has no body.

Due to the structure of basic patterns, the arm is always found in the first row, the leg is always found in the first column, and the body is always found the upper left corner of the matrix. In other words, the arm is the extension of the first row of the body and the leg is the extension of the first column of the body. The following example illustrates the three parts of a UBP.

Example 2.11



Now, we can find all the UBP's for a given number of cells as follows:

1. Find all the bodies; this point is addressed in Section 2.6.2.

2. For each body, the first UBP is constructed by placing all the remaining cells (if any) on the right of the body as an arm. If the arm is full, the remaining cells are placed in the leg.

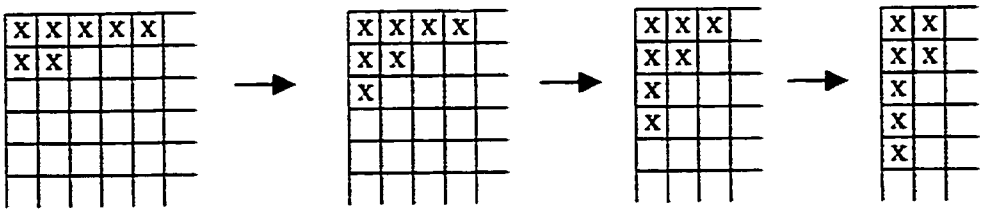
3. The following UBP's for the same body will be generated by moving a cell at a time from the arm and place it in the leg until no more cells remain in the arm or the leg is completely filled.

Obviously, the number of UBP's associated with a body depends on the total number of cells. In case the body uses all the cells (no branches), there will be no UBPs associated to this body except the body itself.

Let us illustrate the above procedure by the following examples:

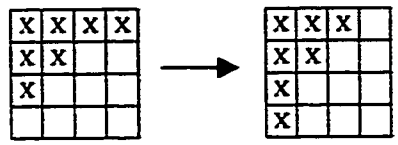
Example 2.12

Consider seven lost cells in any matrix of size ≥ 5×5. Let us generate the UBP's having the smallest body of 4 cells which is the (2×2) body. Following the above procedure for UBP's generation, the UBP's associated with this body are the following four patterns:



Therefore, there are four UBP's of seven cells associated with the 2×2 body in any matrix with size ≥ 5×5.

If the matrix was 4×4 only, the first and last basic patterns in the above example do not fit in the matrix. And consequently, we have only the following two unrecoverable basic patterns constructed from the 2×2 body:



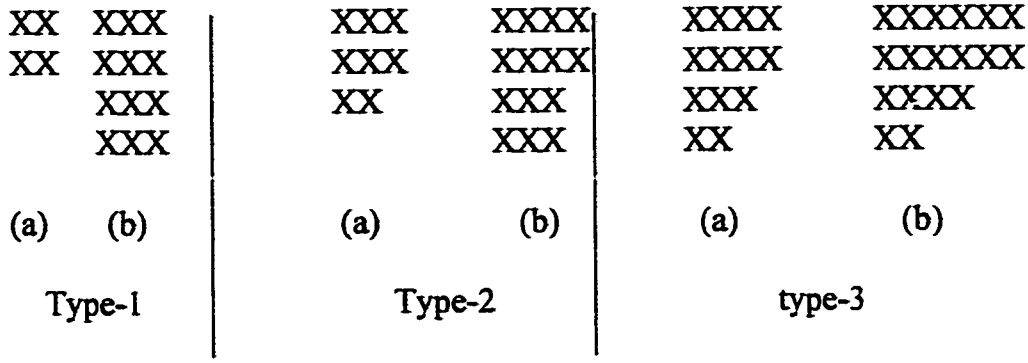
The above procedure demonstrates how to construct all UBP's for a given body. Now, the attention is focused on finding all the bodies. Once this is achieved, all UBP's for a given number of lost cells can be found.

2.6.2 Generating All the Bodies: Algorithmic Approach:

To facilitate the job of enumerating all possible bodies for a given number of lost cells, the bodies will be classified into different types according to the number of different column sizes. We define *type-m* body to be a body with *m* different sizes of columns. That means *type-1* bodies have only one size of columns while *type-2* bodies have two different sizes of columns. Clearly, the first two columns of any body must be of the same size regardless of the size.

Example 2.13

The following are samples of the first three types of bodies.



In Appendix A, the details of the body types idea and an algorithm of constructing them is explained in details. This idea of body types is introduced only to help in building all the bodies and consequently constructing all the UBPs.

Table 2.3 summarizes the number of bodies and UBPs for lost cells ranging from 4 to 18. The details about the shapes of these bodies and the number of UBPs that can be built from each body for each *i*,can be seen in Tables A.1, A.2 and A.3 in Appendix A.

Number of lost cells	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*
Total Bodies	1	1	3	3	6	7	11	13	20	23	33	39	53	65	87
Total UBPs	1	2	5	8	14	21	32	45	65	88	121	160	213	278	367

Table 2. 3 The number of bodies and UBPs for different number lost cells *i*.

Table 2.3 shows that to construct all the UBPs for any number of cells less than 19, we need to construct 87 bodies only. Then for any specific number of lost cells i, out of these 87 bodies all the bodies that consist of i or less cells will be the bodies used to generate all the UBPs for this i. That means, for example, the same 20 bodies used for building the UBPs for 12 cells will be used to build UBPs for the following number lost cells. However, in general any number of cells may need few bodies more than the previous number of cells to build all its UBPs. The way of building the UBPs from bodies was explained in section 2.6.1.

The following table summarizes the results of this chapter.

Number of Lost Cells	Number of RBPs	Number of Bodies	Number of UBPs	Total Basic Patterns	Total Generated Patterns (G)
4	4	1	1	5	1.75E+08
5	5	1	2	7	8.81E+09
6	6	3	5	11	3.69E+11
7	7	3	8	15	1.32E+13
8	8	6	14	22	4.10E+14
9	9	7	21	30	1.13E+16
10	10	11	32	42	2.79E+17
11	11	13	45	56	6.24E+18
12	12	20	65	77	1.27E+20
13	13	23	88	101	2.39E+21
14	14	33	121	135	4.15E+22
15	15	39	160	175	6.69E+23
16	16	53	213	229	1.01E+25
17	15	65	278	293	1.42E+26
18	14	87	367	381	1.89E+27

Table 2. 4: Number of RBPs, Bodies, UBPS and all the possible patterns $\binom{MN}{i}$ for different number of lost cells.

2.7 Summary

In this chapter, we introduced the new concept of basic pattern. Also we showed how to generate the patterns of the same structure of any basic pattern. Finally, for a given i we classify the basic patterns into RBPs and UBPs and give the method of constructing all of them. In the following chapter we will build on the study of this chapter to give two new tight bounds for the value of U_i and consequently the post decoding cell loss rate.

CHAPTER 3

PERFORMANCE BOUNDS FOR POST DECODING CELL LOSS RATE IN PCPC

3.1 Introduction

The results of the novel structural analyses of the lost cell patterns developed in the previous chapter will be used in this chapter to derive three upper bounds on post decoding cell loss rate of the PCPC. Each bound has a degree of tightness in proportion to the degree of complexity. In fact, every bound is obtained by tightening of the previous bound.

To evaluate the post decoding cell loss rate (CLR) for of the parity check product code (PCPC), the number of unrecoverable patterns for i lost cells is required. This is clear from Equation 2.1, which is represented here for convenience.

$$P_d = \frac{1}{MN} \sum_{i=4}^{MN} i U_i P_l^i (1-P_l)^{MN-i} \qquad (3.1)$$

Two approaches for evaluating U_i will be developed in this chapter. The first approach that utilizes the RBP's described in the previous chapter is the basis of the 1st upper bound B-I. In the second approach, B-I is tightened by partially utilizing the UBP's. The resultant bound will be referred to as B-II. In B-II, we will manage to obtain exact evaluation of U_i for some i and close-to-exact for others.

Both bounds are achieved by excluding some patterns that are guaranteed to be recoverable (R_i) from the total number of patterns and assuming the remaining patterns unrecoverable. That is:

$$U_i = \binom{MN}{i} - R_i. \qquad (3.2)$$

The ' i' factor inside the summation in 3.1 is based on the assumption that when a pattern is unrecoverable, all the lost cells will be unrecoverable. The third bound (B-III) is the same as B-II except that this assumption is removed. In B-III the average number of residual unrecoverable cells for the unrecoverable patterns of size i is calculated.

These three bounds are discussed in the following three sections.

These three bounds are discussed in the following three sections.

3.2 First Upper Bound (B-I) :

For i lost cells, let the number of recoverable patterns generated from all the RBPs be V_i . The first bound is obtained by letting $R_i = V_i$. The relation between an RBP and its *generated patterns*, from the recoverability point of view, is given in the following result.

Result 3.1:

If the basic pattern is recoverable, then ***all*** its generated patterns are recoverable.

The proof follows the following argument. In any pattern generated from an RBP all columns, except possibly the first column, contain a single lost cell, and therefore can be recovered by columnwise decoding. The rowwise decoding would then recover all lost cells in the first column, if any.

According to Result 3.1, the value V_i equal to the total number of generated patterns from all RBPs, given by:

$$V_i = \sum_{p=1}^{q_i} G_p \qquad (3.3)$$

where q_i is the total number of RBPs of size i given in Equation (2.9)

In B-I, it is assumed that all patterns generated from unrecoverable basic patterns are unrecoverable. This assumption will be narrowed in developing B-II.

The merit of B-I is its dependence on the RBP's only which can be constructed in a straightforward and computationally simple manner.

Based on the above assumptions, Tables 3.1 to 3.5 show the number of recoverable patterns R_i and the number of unrecoverable patterns U_i and their ratio to the total possible patterns. Recalling that in [14] R_i was assumed to be 0% for $i \geq 6$, a significant improvement can be easily noted in the tables. The improvement becomes clearer for larger matrices. (See Table 3.5).

# OF CELLS i	All possible distributions $\binom{MN}{i}$	Recoverable patterns R_i	$\frac{R_i}{\binom{MN}{i}}$ %	Assumed unrecoverable U_i	$\frac{U_i}{\binom{MN}{i}}$ %
6	7.50E+07	4.8E+07	64%	2.7E+07	36%
7	6.21E+08	2.5E+08	40%	3.8E+08	60%
8	4.43E+09	8.2E+08	19%	3.6E+09	81%
9	2.75E+10	1.7E+09	6%	2.6E+10	94%
10	1.51E+11	2.3E+09	2%	1.5E+11	98%
11	7.44E+11	2.2E+09	0.3%	7.4E+11	99.7%

Table 3.1: R_i and U_i (assuming $R_i = V_i$ only) for 8× 8 matrix.

# OF CELLS i	All possible distributions $\binom{MN}{i}$	Recoverable patterns R_i	$\frac{R_i}{\binom{MN}{i}}$ %	Assumed unrecoverable U_i	$\frac{U_i}{\binom{MN}{i}}$ %
6	3.69E+11	3.2E+11	88%	4.5E+10	12%
7	1.32E+13	1.0E+13	76%	3.2E+12	24%
8	4.10E+14	2.5E+14	60%	1.6E+14	40%
9	1.13E+16	5.0E+15	44%	6.3E+15	56%
10	2.79E+17	8.0E+16	29%	2.0E+17	71%
11	6.24E+18	1.0E+18	17%	5.2E+18	83%
12	1.27E+20	1.1E+19	9%	1.2E+20	91%
13	2.39E+21	9.0E+19	4%	2.3E+21	96%
14	4.15E+22	5.9E+20	1%	4.1E+22	99%

Table 3.2 : R_i and U_i (assuming $R_i = V_i$ only) for 16× 16 matrix.

# OF CELLS i	All possible distributions $\binom{MN}{i}$	Recoverable patterns R_i	$\frac{R_i}{\binom{MN}{i}}$ %	Assumed unrecoverable U_i	$\frac{U_i}{\binom{MN}{i}}$ %
6	1.58E+15	1.5E+15	96%	5.7E+13	4%
7	2.29E+17	2.1E+17	92%	1.8E+16	8%
8	2.92E+19	2.5E+19	86%	4.1E+18	14%
9	3.29E+21	2.6E+21	78%	7.4E+20	22%
10	3.34E+23	2.3E+23	67%	1.1E+23	33%
11	3.08E+25	1.7E+25	56%	1.3E+25	44%
12	2.60E+27	1.2E+27	45%	1.4E+27	55%
13	2.02E+29	7.0E+28	35%	1.3E+29	65%
14	1.46E+31	3.7E+30	25%	1.1E+31	75%
15	9.85E+32	1.7E+32	17%	8.1E+32	83%
16	6.21E+34	7.1E+33	11%	5.5E+34	89%
17	3.68E+36	2.6E+35	7%	3.4E+36	93%
18	2.06E+38	8.6E+36	4%	2.0E+38	96%
19	1.09E+40	2.5E+38	2%	1.1E+40	98%
20	5.48E+41	6.5E+39	1%	5.4E+41	99%

Table 3.3: R_i and U_i (assuming $R_i = V_i$ only) for 32× 32 matrix.

# OF CELLS i	All possible distributions $\binom{MN}{i}$	Recoverable patterns R_i	$\frac{R_i}{\binom{MN}{i}}$ %	Assumed unrecoverable U_i	$\frac{U_i}{\binom{MN}{i}}$ %
6	6.53E+18	6.5E+18	99%	6.5E+16	1%
7	3.82E+21	3.7E+21	98%	8.5E+19	2%
8	1.95E+24	1.9E+24	96%	8.3E+22	4%
9	8.86E+26	8.2E+26	93%	6.4E+25	7%
10	3.62E+29	3.2E+29	89%	4.0E+28	11%
11	1.35E+32	1.1E+32	84%	2.2E+31	16%
12	4.58E+34	3.6E+34	78%	1.0E+34	22%
13	1.44E+37	1.0E+37	71%	4.1E+36	29%
14	4.20E+39	2.7E+39	64%	1.5E+39	36%
15	1.14E+42	6.4E+41	56%	5.0E+41	44%
16	2.91E+44	1.4E+44	48%	1.5E+44	52%
17	6.99E+46	2.9E+46	41%	4.1E+46	59%
18	1.58E+49	5.3E+48	34%	1.0E+49	66%
19	3.40E+51	9.3E+50	27%	2.5E+51	73%
20	6.93E+53	1.5E+53	22%	5.4E+53	78%
21	1.35E+56	2.2E+55	17%	1.1E+56	83%
22	2.49E+58	3.1E+57	13%	2.2E+58	87%
23	4.41E+60	4.1E+59	9%	4.0E+60	91%
24	7.49E+62	5.0E+61	7%	7.0E+62	93%
25	1.22E+65	5.7E+63	5%	1.2E+65	95%
26	1.91E+67	6.1E+65	3%	1.9E+67	97%
27	2.88E+69	6.1E+67	2%	2.8E+69	98%
28	4.19E+71	5.7E+69	1%	4.1E+71	99%

Table 3.4: R_i and U_i (assuming $R_i = V_i$ only) for 64× 64 matrix.

The above four tables are summarized in the following table:

# OF CELLS i	% of Recoverable patterns R_i in 8×8 matrix	% of Recoverable patterns R_i in 16×16 matrix	% of Recoverable patterns R_i in 32×32 matrix	% of Recoverable patterns R_i in 64×64 matrix
6	64%	88%	96%	99%
7	40%	76%	92%	98%
8	19%	60%	86%	96%
9	6%	44%	78%	93%
10	2%	29%	67%	89%
11	0.3%	17%	56%	84%
12	~0%	9%	45%	78%
13	~0%	4%	35%	71%
14	~0%	1%	25%	64%
15	~0%	0.5%	17%	56%
16	0%	0.1%	11%	48%

Table 3.5: Percentage of recoverable patterns R_i (assuming $R_i = V_i$ only) for different sizes of matrices.

It is clear from the above tables that the number of recoverable patterns R_i (assuming $R_i=V_i$ which are generated from RBP's only) is dramatically increasing with increasing the matrix size. For the 64×64 matrix and i<10, 90% or more of the total patterns are recoverable patterns generated from RBPs. The remaining patterns (less than 10%) are generated from UBPs and are assumed unrecoverable. Even when it is possible to identify recoverable patterns generated from UBPs, their effect will not be significant. In other words, for large matrices, B-I is sufficient to give a very good estimation of the performance of the code.

The post decoding cell loss rate based on B-I is depicted in Figure 3.1 for different matrix sizes. It is clear that a smaller matrix performs better than a larger matrix because of the increased redundancy. The redundancy in an 8×8 matrix is 23 %, while it is 12 % in a 16×16 matrix and 6 % in a 32×32 matrix. Figure 3.1 shows that this simple coding scheme is very powerful in recovering lost cells. The post decoding CLR is many orders of magnitude smaller than the pre-decoding CLR.

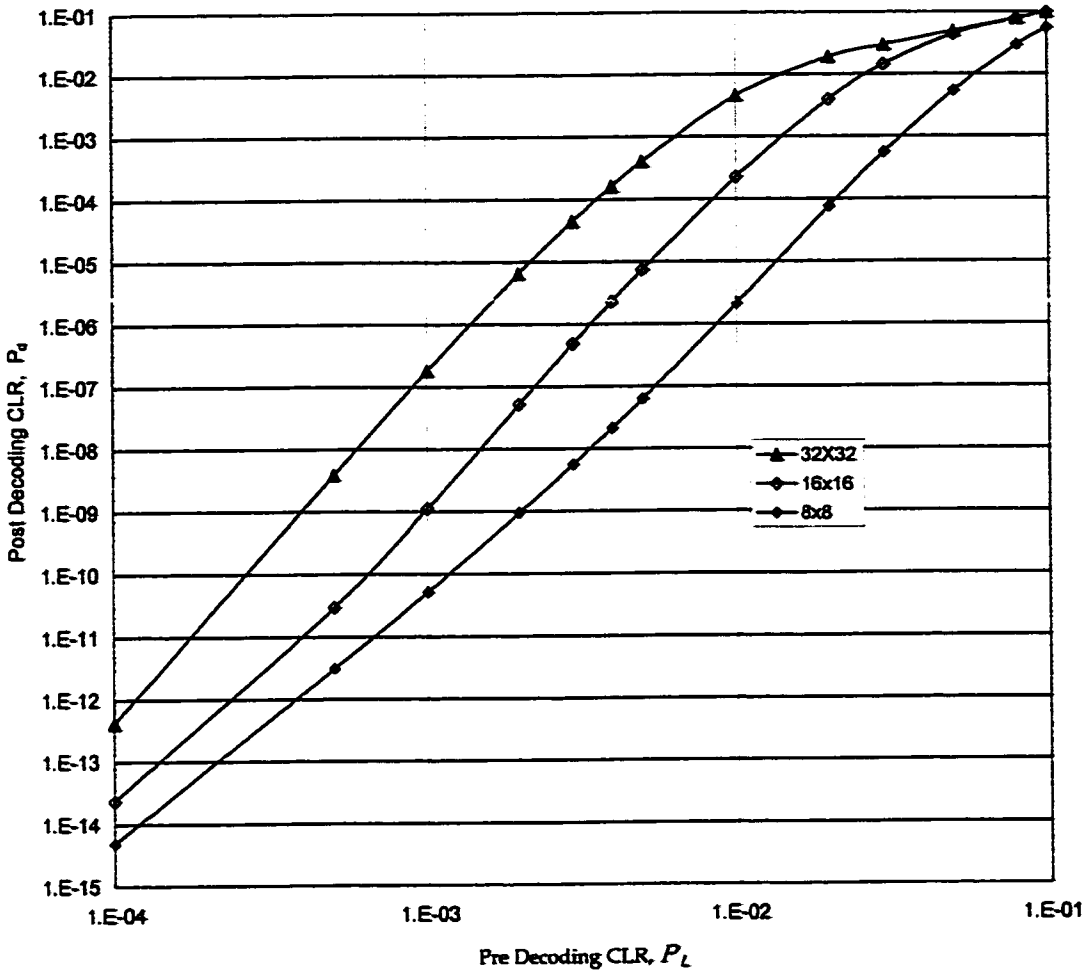


Figure 3.1: Post Decoding CLR using B-I vs. Pre-Decoding CLR for different matrix sizes

For a matrix of size 16×16, the performance of the code using B-I compared to the performance using the bound in [14] is given in Figure 3.2. Note that the two bounds show identical performance at low P_l ($<10^{-4}$) and very high P_l (5×10^{-2}). For low P_l the probability of large number of lost cells $(P_l^i(1-P_l)^{MN-i})$ becomes negligible, which marginelized the accuracy in counting U_i. For large P_l, B-I becomes loose. As a result both bounds exhibit identical performance in these two regions. However in the middle region B-I provides up to about one order of magnitude improvement over the bound in [14].

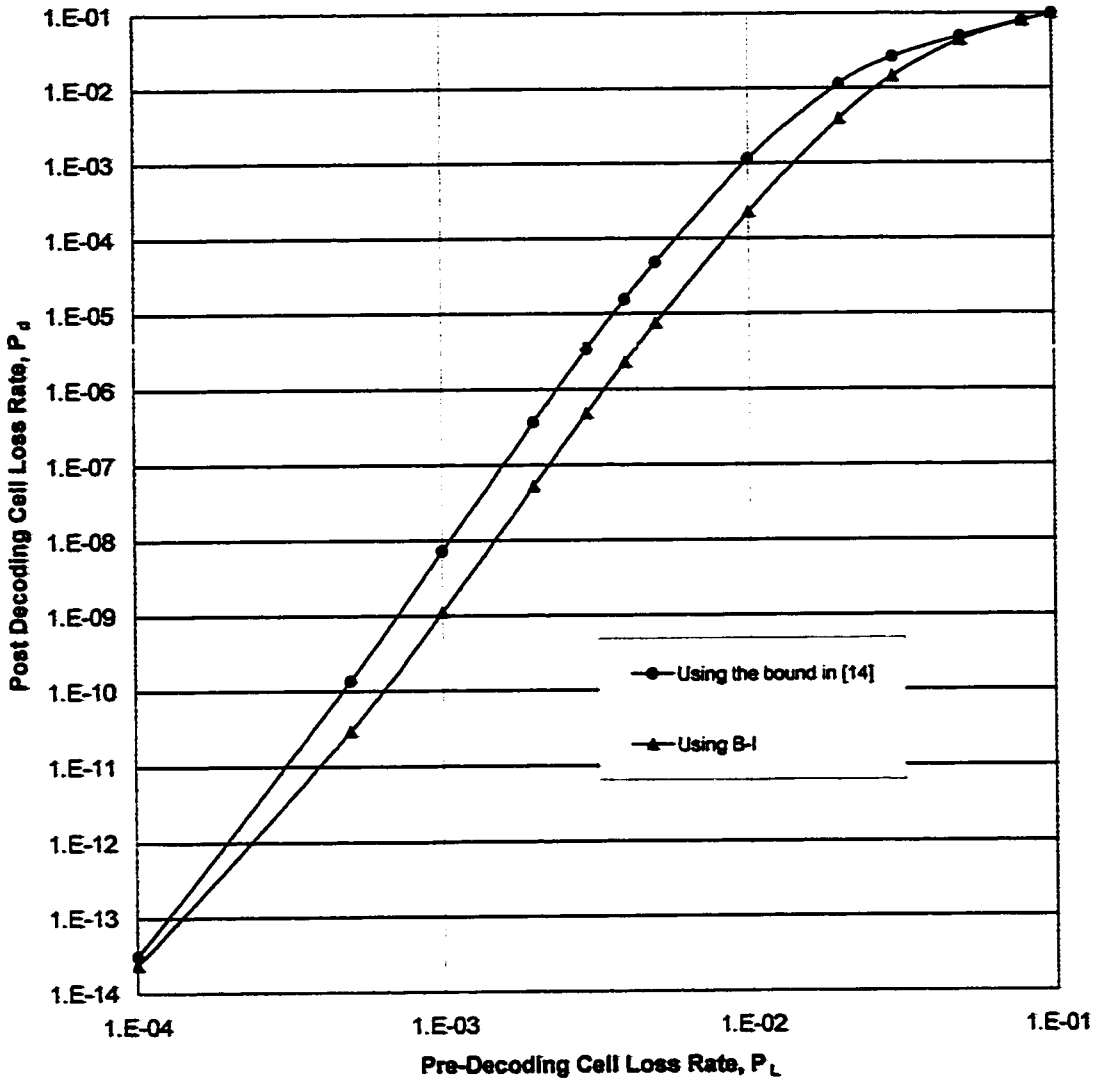


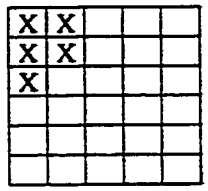
Figure 3.2: Comparison of B-I and [14] for post decoding cell loss rate.

3.3 Second Upper Bound (B-II):

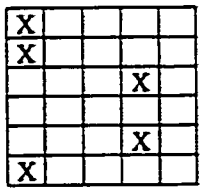
The bound developed in the previous section B-I assumed that all patterns generated from UBPs are unrecoverable. Actually, if a basic pattern is unrecoverable, then its generated patterns could be recoverable or unrecoverable. The following example demonstrate this fact.

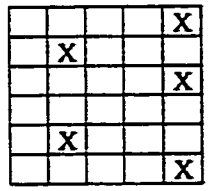
Example 2. 1

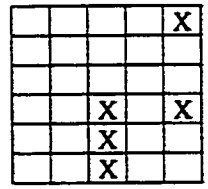
Consider the following unrecoverable basic pattern consisting of five sells:



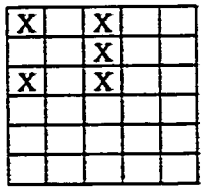
The following three generated patterns are recoverable:

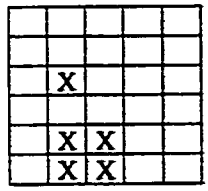


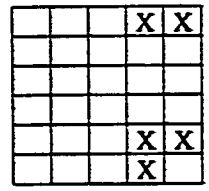




While the following three generated patterns are unrecoverable:







Let W_i be the total number of recoverable patterns which are generated from all the UBPs of size i. If W_i can be evaluated, then a second bound can be obtained by making $R_i = V_i + W_i$. Consequently, the number of assumingly unrecoverable

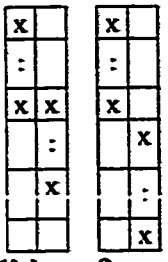
patterns U_i will decrees. Note that no patterns will be assumed unrecoverable while the are actually recoverable (U_i is exact), in case all the recoverable patterns generated from UBPs are identified (i.e. W_i is exact). Finding W_i exactly is difficult for large values of i. This will clear when we discuss how to find W_i in the following sections.

To evaluate W_i, we need to find a way to classify the generated patterns of a UPB into recoverable or unrecoverable patterns. As explained in Section 2.4, there are three different movements for the cells of a basic pattern to produce all its generated patterns. By investigating these types of cell movements, we can see that only the first type of cell movements (distribution over columns) is the one that decides the recoverability of the generated pattern. Moreover, the distribution over columns of the arm cells will not affect the recoverability of the generated patterns. In other words, regardless of their locations, all the cells in the arm are recoverable from the first vertical scan. Therefore, given any UBP the recoverability of the generated patterns depends on the way the cells (excluding the arm cells) are distributed over columns.

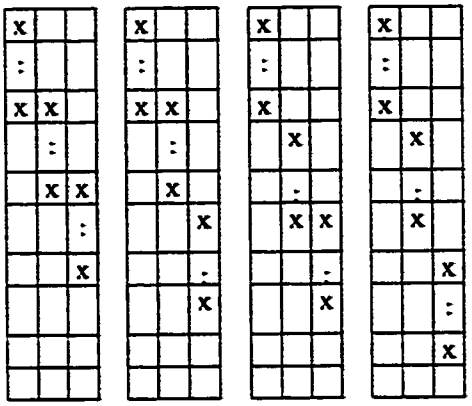
Given a UBP with n occupied columns, out of which a columns are occupied by the arm cells, the number of columns occupied by the body is $t = n - a$. Starting from the second column, if we allow the cells of each column to move column-wise (but they stay adjacent), there will be 2^{t-1} recoverable shapes provided the matrix size is enough for these columnwise moving of the columns. Every column in each one of these shapes has none or one cell overlapping with the preceding columns. In the following example, this is demonstrated for $t = 2, 3$ and 4.

Example 3. 2:

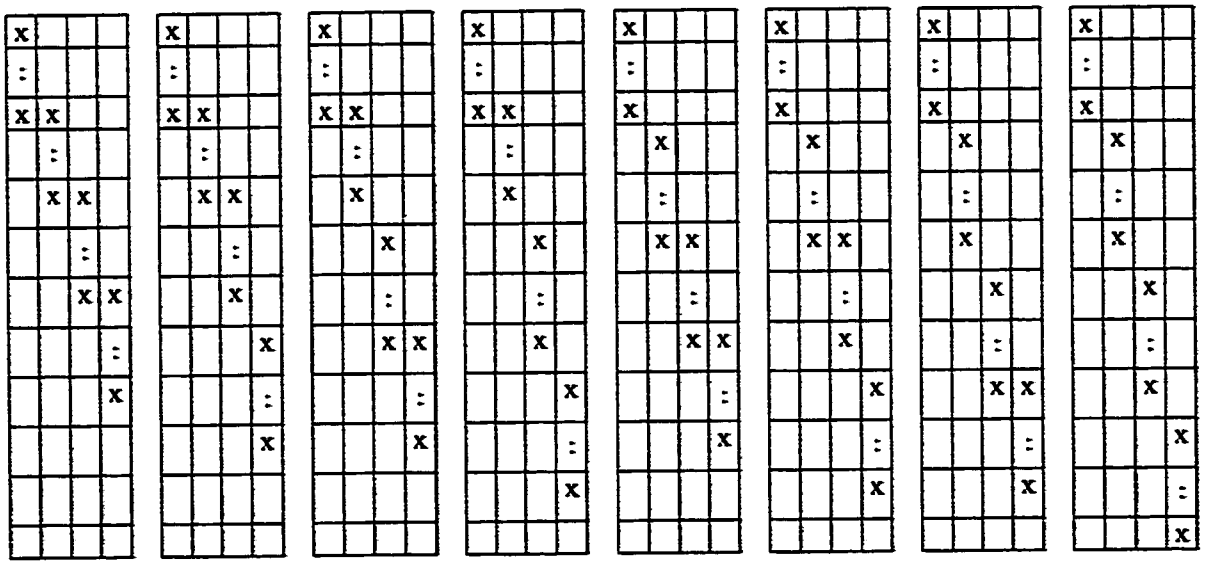
For t=2, these are the two possibilities of none or one cell overlapping:



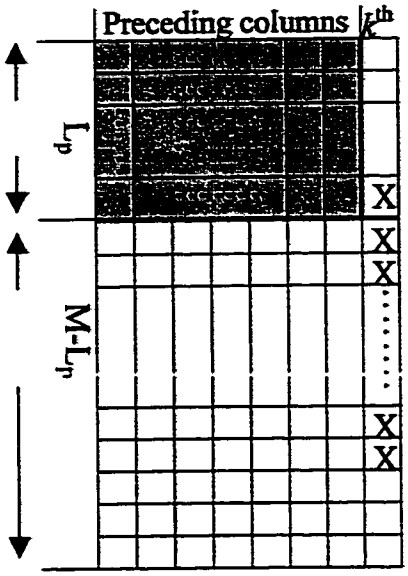
For $t = 3$, these are the four possibilities of none or one cell overlapping:



For t =4, these are the eight possibilities of none or one cell overlapping:

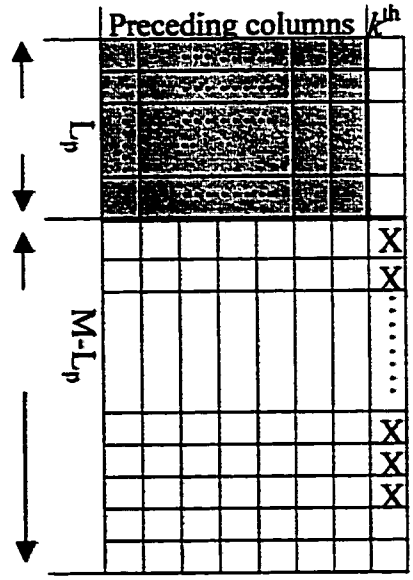


Each of the 2^{t-1} shapes can generate more recoverable shapes by allowing the cells of each column to be distributed over that column preserving its overlapping characteristics with the preceding columns. Let the number of these recoverable distributions for the k^{th} column be d_k where k goes from 2 to t. The value of d_k can be evaluated using one of two expressions depending on whether the k^{th} column has one or no cells overlapping with the preceding columns. Let L_{k-1} to be the total number of rows the first k-1 columns are occupying, and recall that C_k is the number of cells in the k^{th} column. The two expressions of d_k are derived with the aid of Figure 3.3.



The k^{th} column has one overlapping cell with the Preceding columns.

$$d_k = L_{k-1} \times \binom{M - L_{k-1}}{C_k - 1}.$$



The k^{th} column has one overlapping cell with the Preceding columns.

$$d_k = \binom{M - L_p}{C_k}.$$

Figure 3. 3: Evaluating the number of all the recoverable shapes of the k^{th} column.

$$\text{So, } d_k = \begin{cases} L_p \binom{M - L_p}{C_k - 1} & \text{if one cell overlapping with the preceding columns} \\ \binom{M - L_p}{C_k} & \text{if no overlapping with the preceding columns} \end{cases} \quad (3.4)$$

For any of the 2^{t-1} shapes (let it be the j^{th} one), multiplication of all its d_k's for $k =$ 2 to t will yield D_j.

$$D_j = \prod_{k=2}^{t} d_k \quad (3.5)$$

The total number of recoverable patterns generated from the UBP under investigation (r_p)is:

$$r_p = \binom{M}{C_1} \times \sum_{j=1}^{2^{t-1}} D_j \times M^a \times T_{p,1} \times T_{p,2} \quad (3.6)$$

where a is the number of cells in the *arm* and $T_{p,1}$ and $T_{p,2}$ are as defined in Section 2.4.

For i lost cells, let the number of UBPs be h_i .The total recoverable patterns generated from all UBPs be W_i.

$$W_i = \sum_{p=1}^{h_i} r_p \quad (3.7)$$

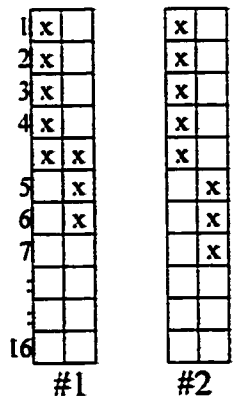
Example 3. 3

For any UBP with t=2, j takes the value 1 or 2 . The value of $D_j = d_2$ for the corresponding shape.

For one cell overlapping shapes: $d_2 = C_1 \times \binom{M - C_1}{C_2 - 1}$.

For no overlapping shapes : $d_2 = \binom{M - C_1}{C_2}$

As a particular example, let C1=5 ,C2=3 and M=16 (as shown below)



For shape #1: $D_1 = d_2 = 5 \times \binom{16-5}{2} = 275$.

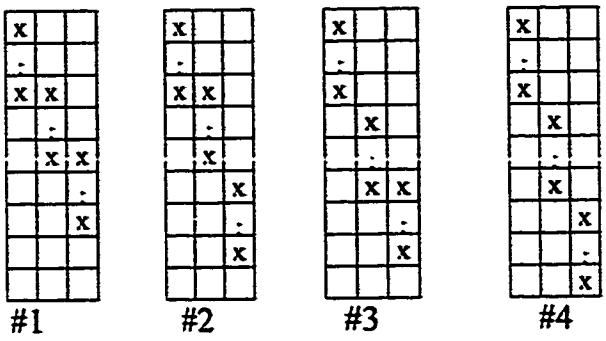
For shape #2 : $D_2 = d_2 = \binom{16-5}{3} = 165$.

D1+D2=440.

That means out of all the possible distributions of the second column cells (which equal $\binom{16}{3} = 560$), 440 distributions are recoverable.

Example 3. 4

For any UBP with t=3, D_j=$d_2 \times d_3$ for j=1 to 4. In the following we use (3.4) to evaluate d_2 and d_3 for each of the four possible shapes.



$d_2 = C_1 \times \binom{M - C_1}{C_2 - 1}$ for the first two shapes,

while $d_2 = \binom{M - C_1}{C_2}$ for the last two shapes

Note that two d_2's are the same when t=2.

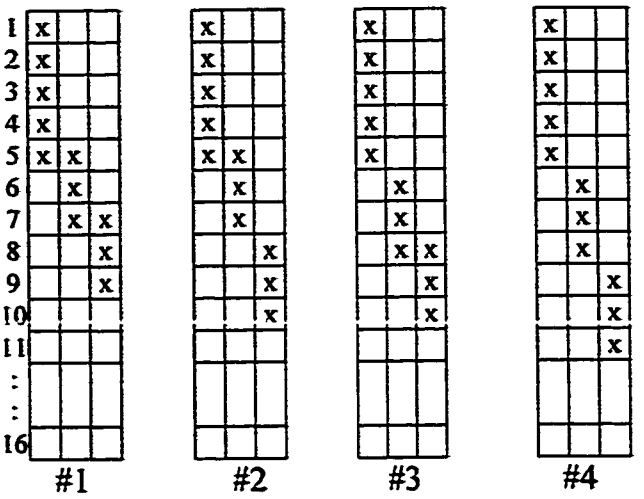
For #1, $d_3 = (C_1+C_2-1) \times \binom{M - (C_1 + C_2 - 1)}{C_3 - 1}$

For #2, $d_3 = \binom{M - (C_1 + C_2 - 1)}{C_3}$

For #3, $d_3 = (C_1+C_2) \times \binom{M - (C_1 + C_2)}{C_3 - 1}$

For #4, $d_3 = \binom{M - (C_1 + C_2)}{C_3}$

As particular example let C_1=5,C_2=3,C_3=3 and M=16:



$$D_1=d_2 \times d_3 = 275 \times 7\times\binom{16-7}{2} = 69300.$$

$$D_2=d_2 \times d_3 = 275 \times \binom{16-7}{3} = 32100.$$

$$D_3=d_2 \times d_3 = 165 \times 8\times\binom{16-8}{2} = 36960.$$

$$D_4=d_2 \times d_3 = 165 \times \binom{16-8}{3} = 9240.$$

$D_1+D_2+D_3+D_4 = 147600.$

The total possible distributions of the 2ed and the 3rd columns $= \binom{16}{3}\binom{16}{3}$ which is equal to 313600; out of these 147600 are definitely recoverable and the remaining are assumed unrecoverable.

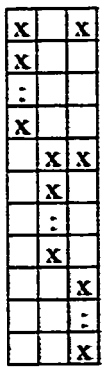
In (3.6), we assumed that the 2^{t-1} recoverable shapes having none or one cell overlapping are the only recoverable shapes produced by allowing the cells of each column to move column-wise (but they stay adjacent). This assumption is correct only for t=2. Starting from t=3, there will be some shapes consisting of column(s) having more than one cell overlapping but they are still recoverable. If we call the number of these special shapes s, then (3.6) will be:

$$r_p=\binom{M}{C_1}\times\sum_{j=1}^{s+2^{t-1}}D_j\times M^a\times T_1\times T_2 \qquad (3.8)$$

For these special shapes, d_k does not follow 3.4 and must be evaluated in a different way.

There is no special shapes for t=2. That implies that for any UBP having 2-column body, the number of recoverable patterns r_p generated using (3.6) is exact; and all the remaining patterns generated from this UBP definitely unrecoverable.

For t=3, the following is the only special shape in which the third column has two cells overlapping with he preceding columns but the shape is still recoverable.



For this shape, d_2 is the same as the third and the fourth standard shapes in Example 3.4. However d_3 has a different expresion which can be easily derived:

$$d_3 = C_1 \times C_2 \times \binom{M-(C_1+C_2)}{C_3-2}. \qquad (3.9)$$

By taking this special shape into consideration, the number of recoverable patterns generated from UBPs with t=3 can be evaluated exactly.

Unfortunately, it seems to be that constructing these special shapes can not be done in a systematic way. Because of that we assume that all the special shapes are unrecoverable for $t \geq 4$. This is the only assumption in calculating the number of recoverable patterns from all the UBPs, W_i. This implies that for 4,5,6 and 7 lost cells, the value of W_i is exact. For 8 lost cells, the error in W_i is negligible (less than 10^{-6} %). The error continue increasing with increasing i and then start decreasing again when i gets closer to 32.

In general, the percentage of unrecoverable patterns in $M \times N$ matrix starts very small at small number of cells (starting from 4), and then continue increasing. It reach almost 100% of the patterns when the number of lost cells get close to M+N. All the patterns will be unrecoverable for $i \geq M+N$. The percentages of W_i and V_i are shown in Table 3.6 for different i in 16×16 matrix. Figure 3.4, is chart showing the percentages of recoverable patterns for different number of lost cells.

It is easy to note that for small values of i the majority of recoverable patterns are generated from RBPs while the recoverable patterns generated from UBPs are more than those generated from RBPs for $i>9$. In general, up to 13 lost cells, more than 90% of the patterns are recoverable. This is a considerable improvement compared to [14] where it was assumed that no recoverable patterns for any $i>5$.

Number of cells	Number of Possible Patterns	Percentage of Recoverable Patterns Generated from RBPs (V_i %)	Percentage of Recoverable Patterns Generated from UBPs (W_i %)	Percentage of Definitely Recoverable Patterns ($V_i + W_i$) %
4	1.7E+08	99%	1%	100%
5	8.8E+09	95%	5%	100%
6	3.7E+11	88%	12%	100%
7	1.3E+13	76%	24%	100%
8	4.1E+14	60%	39%	99%
9	1.1E+16	44%	55%	99%
10	2.8E+17	29%	69%	98%
11	6.2E+18	17%	80%	97%
12	1.3E+20	9%	86%	95%
13	2.4E+21	4%	88%	92%
14	4.1E+22	1%	87%	88%
15	6.7E+23	0%	82%	83%
16	1.0E+25	0.12%	76%	76%
17	1.4E+26	0.03%	67%	67%
18	1.9E+27	0.01%	58%	58%
19	2.4E+28	~0%	47%	47%
20	2.8E+29	~0%	36%	36%
21	3.2E+30	~0%	27%	27%
22	3.4E+31	~0%	18%	18%
23	3.4E+32	~0%	12%	12%
24	3.3E+33	~0%	7%	7%
25	3.1E+34	~0%	4%	4%
26	2.7E+35	~0%	2%	2%
27	2.3E+36	~0%	1%	1%
28	1.9E+37	~0%	~0%	~0%
29	1.5E+38	~0%	~0%	~0%
30	1.1E+39	~0%	~0%	~0%
31	8.3E+39	~0%	~0%	~0%
>31		0%	0%	0%

Table 3. 6: Percentage of V_i, W_i and total Recoverable patterns for different number of lost cells i in a 16× 16 matrix.

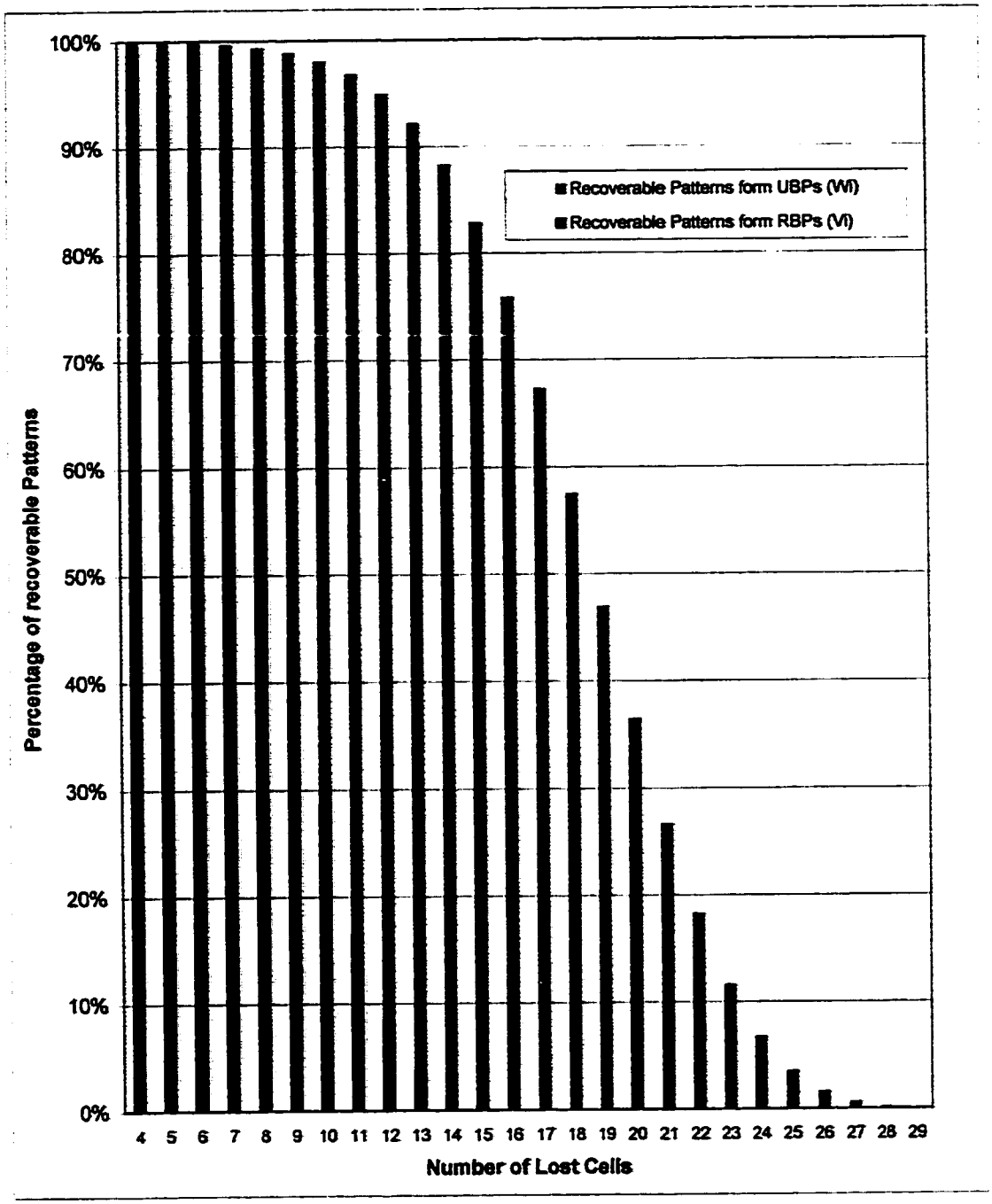


Figure 3.4: Percentage of Recoverable Parterres in 16×16 matrix for different number of cells.

Now $R_i = V_i + W_i$ and consequently 3.2 will be:

$$U_i = \binom{MN}{i} - V_i - W_i . \qquad (3.10).$$

Substituting this U_i in Equation 3.1 yields B-II for post decoding CLR (P_d). The plots of P_d vs. the pre-decoding cell loss rat P_l using B-I, B-II and the bound in [14] are shown in Figure 3.5. The improvement introduced by B-II is evident. B-II yields a lower post decoding CLR over a wide range of P_l. This improvement reaches up to 2.5 orders of magnitude over that in [14] at $P_l = 10^{-2}$.

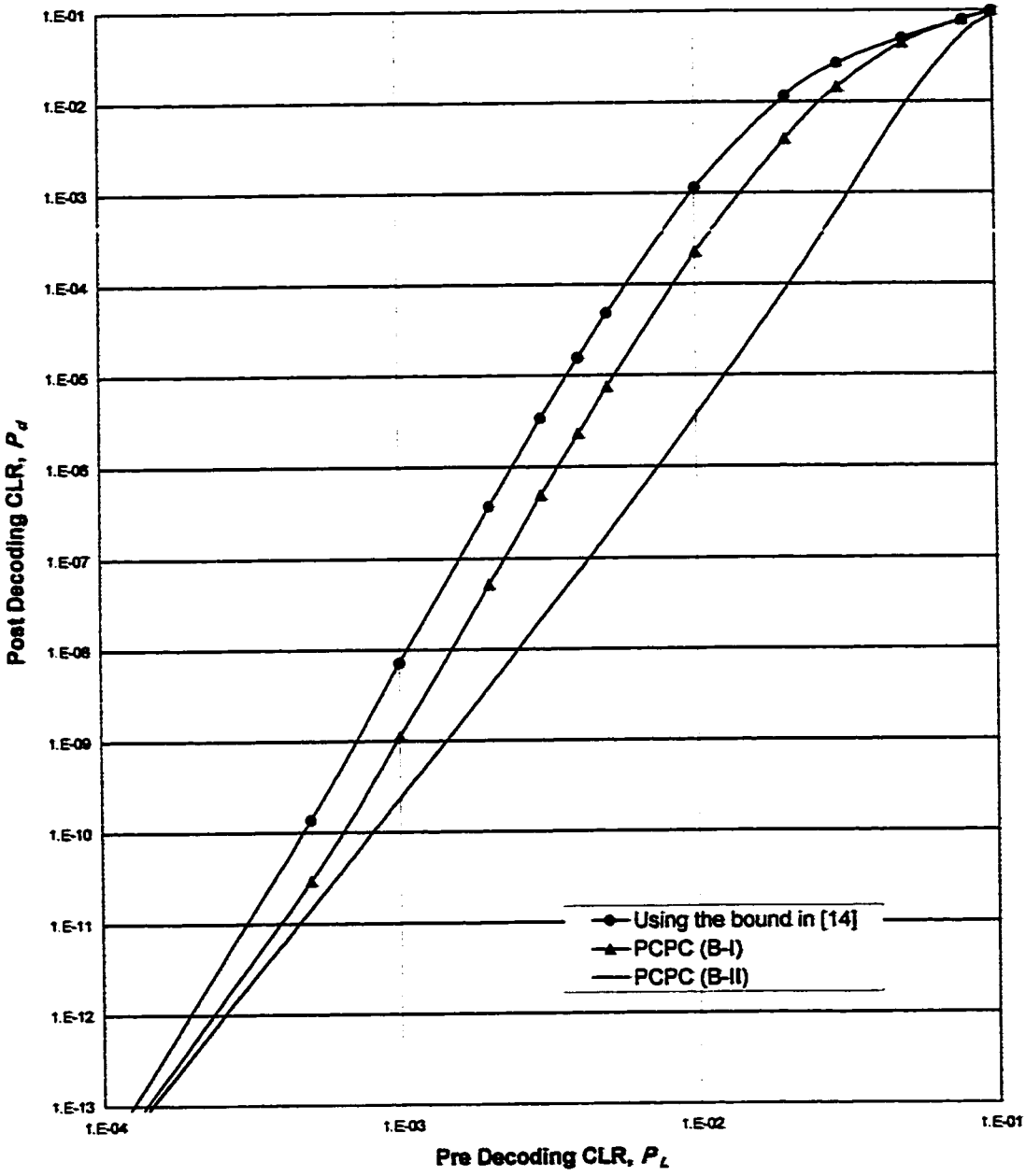


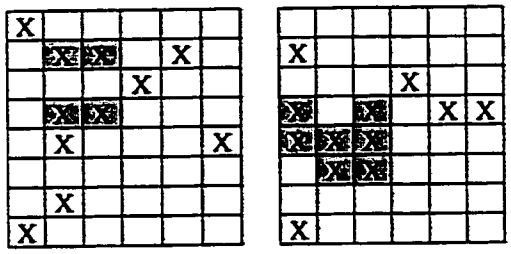
Figure 3.5: the Performance of the PCPC using the bound in [14], B-I and B-II .

3.4 The Third Upper Bound (B-III) :

In [14], B-I and B-II it was assumed that when a pattern is unrecoverable all its cells are unrecoverable. In other words, the number of cells in an unrecoverable pattern is the same before and after decoding. Clearly some lost cells in unrecoverable pattern may be recovered. B-II can be further tightened by considering only the number of unrecoverable cells in an unrecoverable pattern after decoding.

In any decoding iteration any cell that is found alone in a column or a row can be recovered. Therefore, all arm cells can be recovered. Also, many other cells in the body and the leg can be recovered.

Consider for example, the following two unrecoverable pattern:



In the first shape, all the cells can be recovered except the four shaded cells . Similarly, In the second shape, all the cells can be recovered except the seven shaded cells

For any unrecoverable patterns, an algorithm was set up utilizing some concepts (like: arm, leg, bodies and others) find the average number of unrecoverable cells in an unrecoverable pattern. For the total unrecoverable

patterns of size i, let the average number of lost cells after decoding be e_i. The results of the algorithm are shown in Table 3.7.

Number of Cells in Unrecoverable patterns i	Average Number of Cells After Decoding e_i	$\frac{e_i}{i}$ %
4	4	100%
5	4	80%
6	4.0006	67%
7	4.002	57%
8	4.06	51%
9	4.2	47%
10	4.6	46%
11	5.3	48%
12	6.2	52%
13	7.2	56%
14	8.3	60%
15	9.5	63%
16	10.5	66%
17	11.5	68%
18	12.5	70%
19	13.5	71%
20	14.4	72%
21	15.4	73%
22	16.4	75%
23	17.5	76%
24	18.5	77%
25	19.6	79%
26	20.8	80%
27	21.9	81%
28	23.0	82%
29	24.2	83%
30	25.4	85%
31	26.5	86%
32	27.7	87%

Table 3. 7: Average number of cells after decoding *(e_i) for* i-cell unrecoverable patterns.

The post decoding cell loss rate will be modified to be:

$$P_d = \frac{1}{MN} \sum_{i=4}^{MN} e_i U_i P_l^i (1-P_l)^{MN-i} \qquad (3.11)$$

The difference between Eq. (3.1) and Eq. (3.11) is the use of e_i instead of i.

The extra improvement introduced by B-III is shown in Figure 3.6. The curves produced using B-II and B-III are identical for low P_l for the same reason mentioned in the previous bounds. At high P_l , both curves are also identical because most of the cells in large size patterns are unrecoverable.

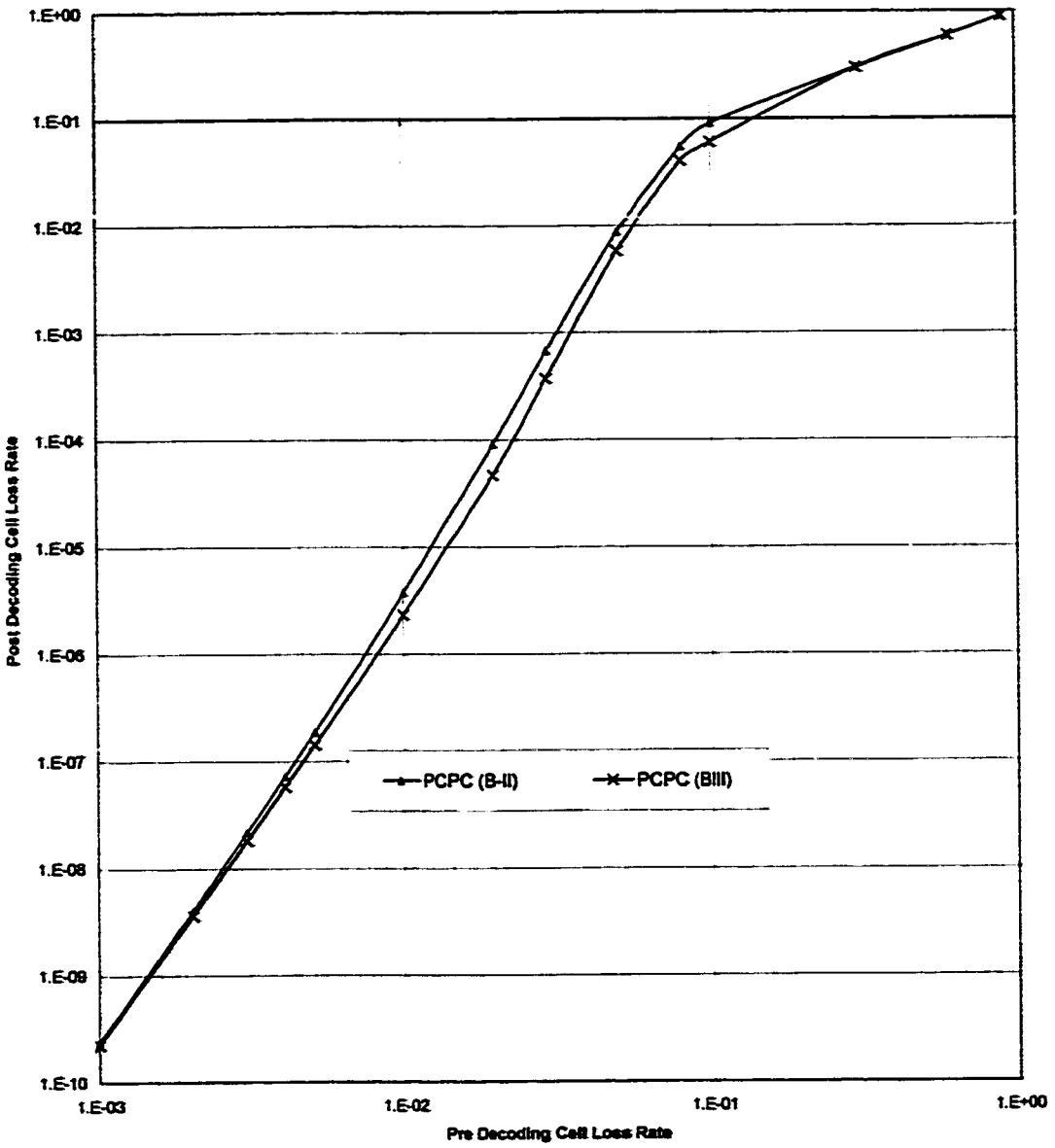


Figure 3.6 : The performance of the code assuming all the cells of unrecoverable patterns are unrecoverable (using *i)* and by recovering the recoverable cells of unrecoverable pattern (using e_i).

B-III is the tightest of the three bounds. To test the degree of tightness of B-III, the 16×16 was simulated. Comparison between simulation and analytical work is shown in Figure 3.7. The Figure shows that B-III is almost exact. Therefore, the developed approach of structural analysis provides us with a true measure of the capability of the PCPC.

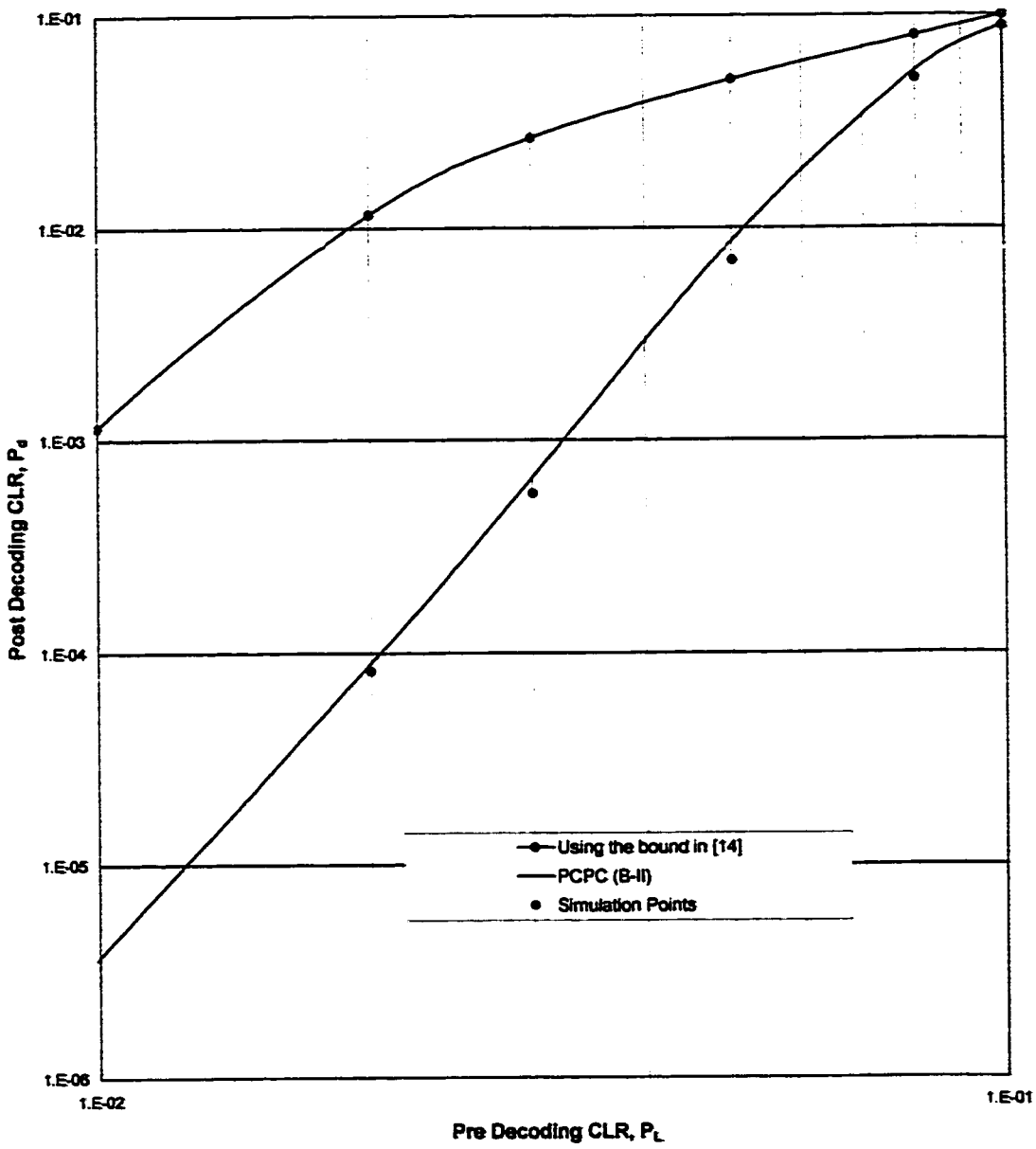


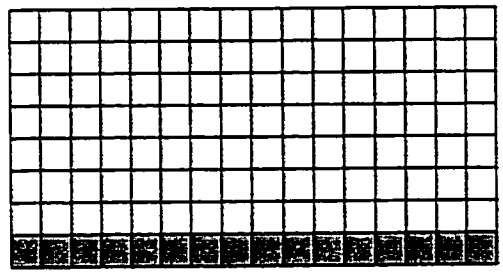
Figure 3.7: The performance of the code compared to some simulation points.

3.5 Comparison of PCPC to other coding schemes:

The PCPC analyzed in this study is compared with two coding schemes. The first scheme is the one dimensional parity check code proposed in [15]. The scheme is depicted in Figure 3.8 For the sake of fair comparison the dimension of matrix for the one dimensional code is taken to be 8×16 so that it has almost the same redundancy rate of PCPC of size 16×16.

In the one dimensional parity check code, one lost cell per column can be recovered, but two or more lost cells lead to an unrecoverable cell loss pattern. Therefore the post decoding cell loss rate will be:

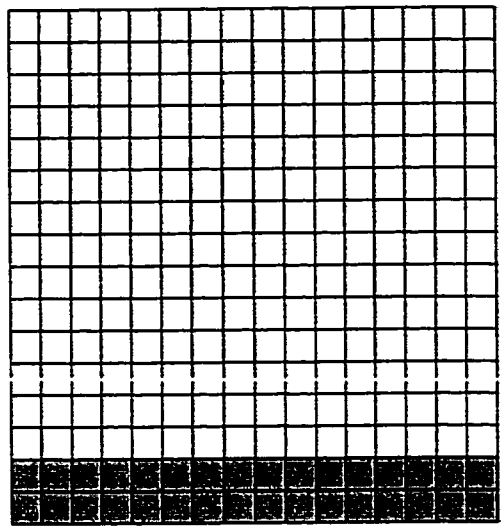
$$P_d = \frac{1}{M}\sum_{i=2}^{M} i\binom{M}{i}P_l^i(1-P_l)^{M-i} \qquad (3.12)$$



■ Represents the parity check cells.

Figure 3. 8: One Dimensional Parity Check Code.

The authors in [15] also suggested to use RS code instead of the parity check code to increase the power of recovering lost cells. The suggested RS code for loss cell recovery having the same redundancy as the PCPC is shown in Figure 3.9.



■ Represents the parities of an RS code.

Figure 3. 9: The RS Erasure code proposed for comparison.

An RS code with α parity symbols can recover α erasures [5]. For the RS code in Figure 3.9, one or two lost cell per column can be recovered, but three or more lost cells lead to unrecoverable cell loss pattern. Therefore the post decoding cell loss rate will be:

$$P_d = \frac{1}{M}\sum_{i=3}^{M} i\binom{M}{i}P_l^i(1-P_l)^{M-i} \qquad (3.13)$$

The performance of the three codes are shown in 3.10. It is clear that PCPC gives much better performance than both codes over wide range of pre-decoding cell loss rate. At very high P_l, all codes give the same performance. Around 8×10^{-2} the RS code gives little bit better performance than the other two codes.

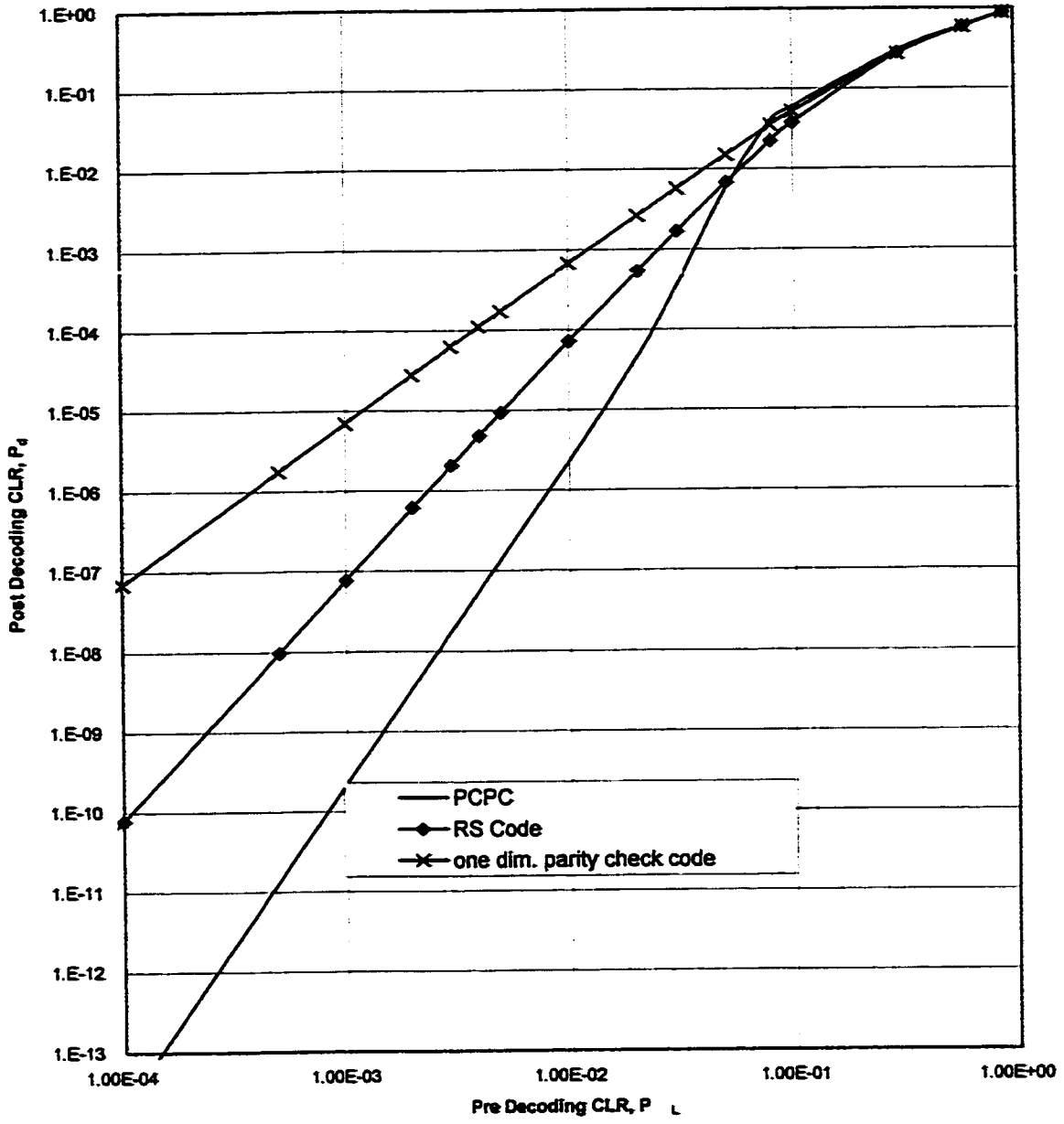


Figure 3.10: Comparison between the PCPC, One dim. Parity Check Code and RS Code

3.6 Summery

In this Chapter, we derived three bounds for the post decoding CLR using the PCPC. Each bound has a degree of tightness in proportion to the degree of complexity. In fact, every bound is obtained by tightening of the previous bound. From that result Bound III can be considered the ultimate result of this part of the work, with Bound I and Bound II being intermediate steps of derivation. The system was simulated to insure the accuracy of the calculated performance.

CHAPTER 4

HYBRID ARQ/PCPC FOR ATM NETWORKS

4.1 Introduction:

One of the most serious problems in ATM networks is that ATM cells may be discarded during the transmission. Cell discarding seriously degrades transmission quality. Because high quality and reliability are essential in B-ISDN/ATM networks, the employment of some error control schemes to improve the cell loss rate is necessary. In this Chapter we will analyze the performance of different error control schemes to recover lost ATM cells transmitted between two

points (nodes) through a specific link. Figure 4.1 shows the general model for this study.

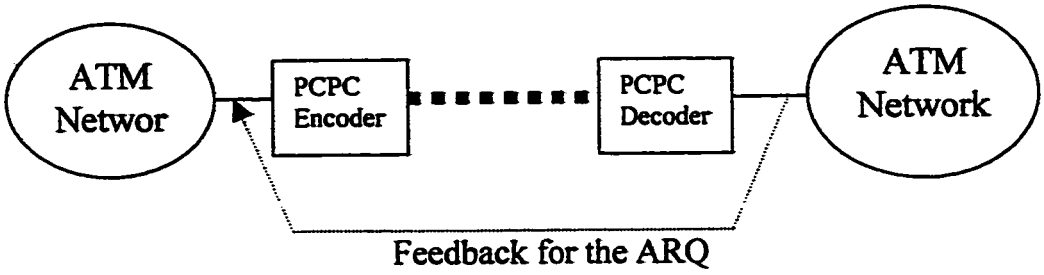


Figure 4.1: Configuration of the proposed Hybrid ARQ/PCPC model.

Three error control schemes will be considered. Firstly, we will investigate the use of the PCPC and evaluate its performance for different channel bit error rate (BER). Secondly, we will study the performance of pure ARQ system to recover lost cells based on the Selective Repeat (SR) protocol. Two retransmission strategies will be considered. In the first one, if the matrix contains any lost cells the whole matrix is retransmitted. This strategy is similar to that in [17]. In the second strategy, only lost cells are retransmitted. Finally, we will analyze the performance of a hybrid ARQ/PCPC coding scheme for recovering ATM cells.

4.2 Cell Loss Rate:

For the model in Figure 4.1, a cell is discarded if its header is detected to be in error. Every ATM cell has a four-bit Serial Number which can be used for cell loss detection. Since SN can represent 16 numbers, up to 16 consecutive lost cells

can be detected by checking the continuation of the serial number. For this reason we will apply the code over a matrix of 16×16 ATM cells. This size of matrix was also adopted in [14-17].

Let p be the BER of the link. Because the header consists of 40 bits, the probability, P_c, that the header is correct is:

$$P_c = (1-p)^{40} \qquad (4.1)$$

Consequently, the cell loss rate (P_l) is

$$P_l = 1-(1-p)^{40} \qquad (4.2)$$

In deriving 4.2 it is assumed that all header error patterns are detectable. Considering the power of the CRC error detection code, this assumption is well justified. The same assumption was made in [15] and [17].

Using 4.2, the post decoding cell loss rate (P_d) is:

$$P_d = \frac{1}{MN}\sum_{i=4}^{MN} e_i U_i \left[1-(1-p)^{40}\right]^i \times \left[(1-p)^{40}\right]^{MN-i} \qquad (4.3)$$

Where: MN is the matrix size.

i is the number of pre-decoding lost cells.

e_i is the average number of lost cells in an unrecoverable pattern after decoding for i lost cells.

U_i is the total number of unrecoverable patterns of size i.

p is the link bit error rate.

The post decoding cell loss rate versus bit error rate is shown in Figure 4.2.

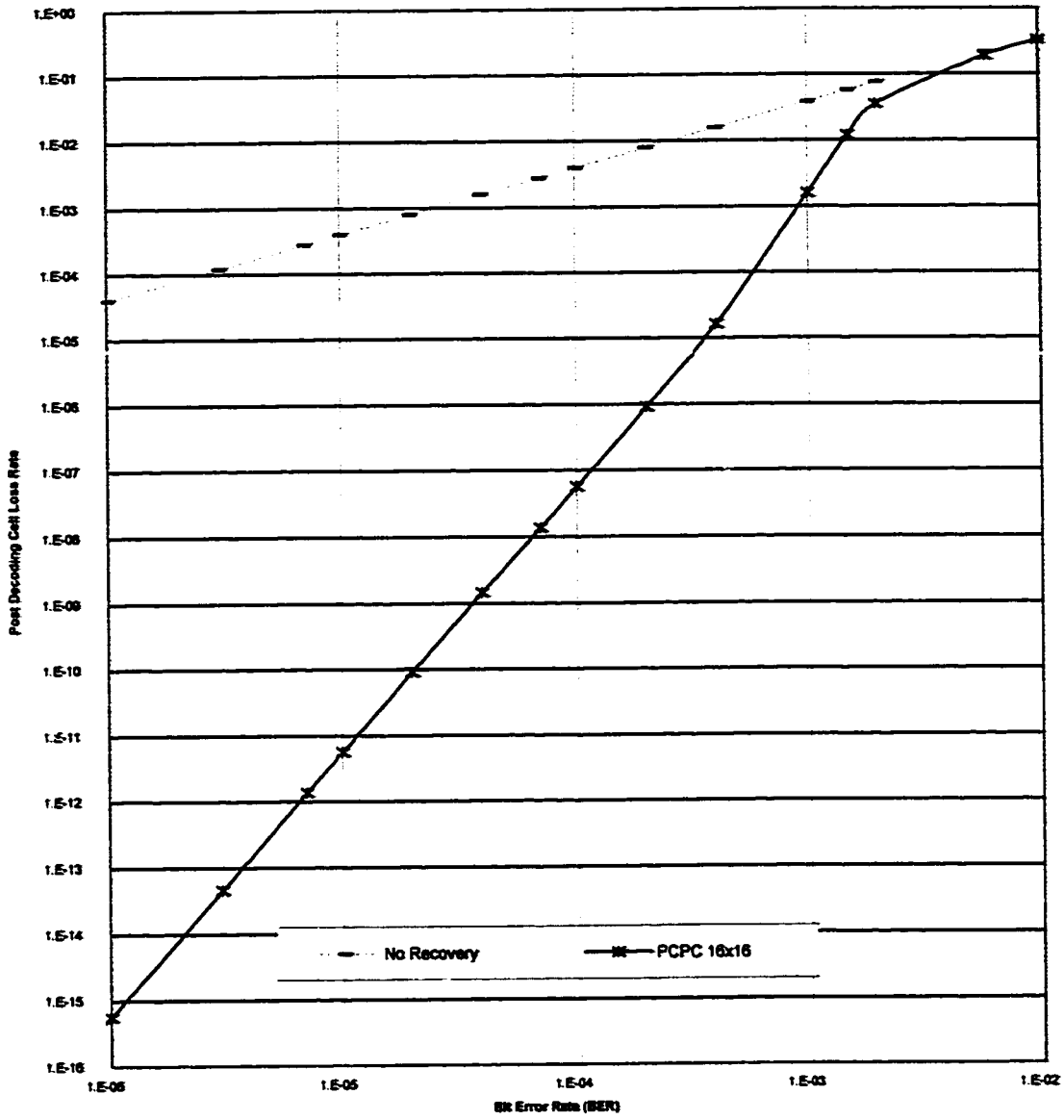


Figure 4.2: Post Decoding CLR vs. Bit Error Rate of the ATM link.

4.3 Pure ARQ Scheme:

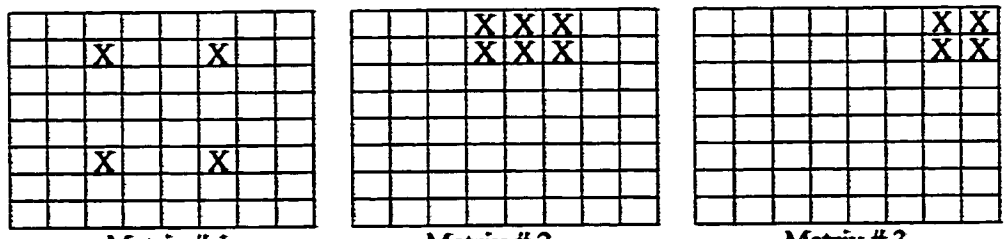
Here we analyze a pure ARQ scheme when lost cells are recovered by retransmission. The analysis is based on the Selective Repeat ARQ (SR-ARQ). It is assumed that the buffer size is infinite and the feedback channel is error free.

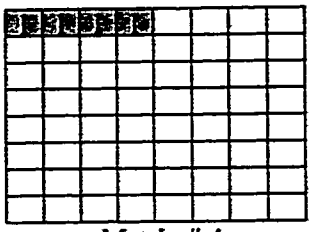
Two retransmission strategies are investigated. In the first strategy the full matrix is requested for retransmission if it contains one ore more lost cells. In the second strategy, the system requests for the lost cells only.

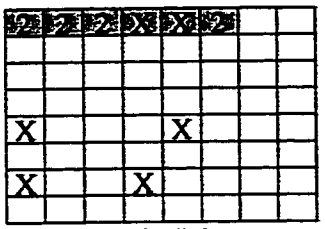
Requesting retransmission of the lost cells only instead of the full matrix is expected to improve the throughput efficiency. However, a careful consideration must be given to two implementation issues, retransmission mechanism and the acknowledgment mechanism.

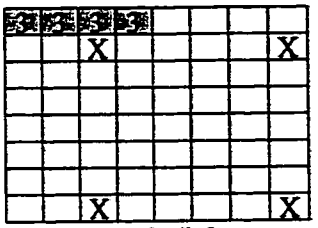
Retransmission mechanism:

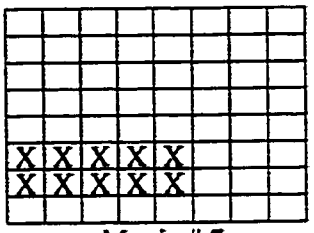
Let's assume that the round trip delay for the ARQ scheme is equivalent to transmission time of an integer number of matrices, say j. If the ARQ system receives the i^{th} matrix containing k lost cells, the retransmitted cells will be the first k cells in the $(i+j)^{th}$ matrix. This retransmission mechanism is illustrated below for j=3.

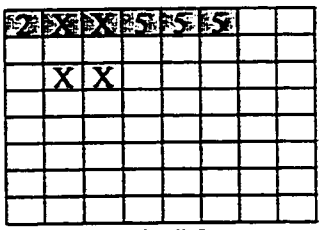


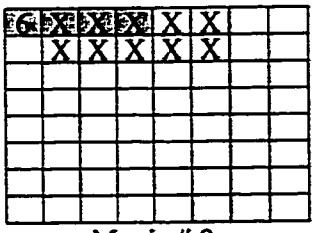

Matrix # 4

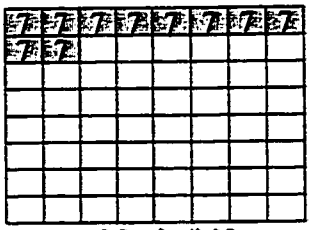

Matrix # 5

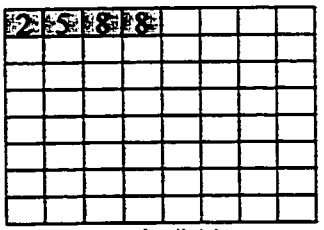

Matrix # 6

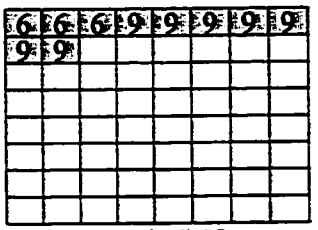

Matrix # 7


Matrix # 8


Matrix # 9


Matrix # 10


Matrix # 11


Matrix # 12

Where: ■ Represents a repeated cell and the number inside it indicates the original matrix of the cell.

□ Represents a new cell.

X Represents a lost cell.

For example, Matrix 3 is received by the ARQ system containing four lost cells. A request is made for these four cells to be retransmitted (see next section for acknowledgment). The transmitter receives the request and accordingly, transmits

the cells in the first four locations of Matrix 6. When Matrix 6 is received, the first four cells are transferred to fill Matrix 3.

Acknowledgment:

For each matrix, one ATM cell is sent back as an acknowledgment. To support the assumption that the feedback is error free, a very powerful protection code with a rate of ½. is used on the feedback link. The proposed format for the acknowledgment cell is shown below.

23 bytes	23 bytes	1 byte	1 byte
PB	AD	NL	PN

Where:

PB : Parity bits of a (46,23) code.

AD : Acknowledgment data or Address of the lost cells.

NL : Number of lost cells.

PN : Parity bits of a code used to protect NL.

Figure 4.3: Format of the Acknowledgment Cell

Each byte of AD indicates the location of one lost cell in the matrix (0 to 255). Up to 23 lost cells, can be negatively acknowledged. In the unlikely case of more than 23 lost cells, the whole matrix is requested for retransmission. NL indicates the

number of lost cells lost cells in the matrix. In other words it indicates how many useful bytes can be found in AD. The remaining bytes of AD are not used and can be filled by zeros. NL=0 indicates that there are no lost cells and it is considered as a positive acknowledgment for the matrix.

It is worth mentioning that a cell with none zero NL is considered a NACK for the cells specified in the AD, and a ACK for the remaining cells in the matrix. Therefore, when an acknowledgment cell is received back, the cells that are not indicated in AD can be released from the buffer.

The acknowledgment mechanism is illustrated in Figure 4.4.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56		58		60	61	62	63
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88		90		92	93	94	95
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

a) Received matrix

Parity (23 bytes)	57	59	89	91	0 padding (19 bytes)	4	PN

b) Acknowledgment cell

Figure 4.4: Acknowledgment Mechanism

Throughput Efficiency:

The performance of ARQ systems is evaluated by studying the throughput efficiency or simply throughput. The throughput (η) is the ratio of the average number of accepted data cells to the number of transmitted cells in a unit of time. When SR-ARQ is used the throughput is given by [5]:

$$\eta = \frac{1}{N_T} \times \frac{k}{n} \qquad (4.4)$$

where N_T is the average number of transmissions required to successfully receive the block of data under investigation (the full matrix or one cell in our case).

In the ARQ scheme explained above, cell detection is made using the serial number which implies no redundancy for cell loss detection (k/n=1). Therefore the throughput is given by:

$$\eta_A = \frac{1}{N_T} \qquad (4.5)$$

Let us evaluate the throughput when the whole matrix is requested for retransmission if it contains any lost cells (first strategy).

Define P and P_e such that:

$P=(1-P_l)^{256}$ is the probability of no lost cells in the matrix.

$P_e = 1-P$ is the probability that the matrix contain one or more lost cells.

The probability to receive the full matrix successfully by one transmission = P.

The probability to receive the full matrix successfully after two transmissions = PP_e.

The probability to receive the full matrix successfully after three transmissions = $P(P_e)^2$.

The probability to receive the full matrix successfully after t transmission = $P(P_e)^{t-1}$.

Therefore, the average number of transmissions required to receive the full matrix successfully:

$$N_T = [1 \times P] + [2 \times P \times P_e] + [3 \times P \times P_e^2] + [4 \times P \times P_e^3] + \ldots\ldots$$

Which can be rewritten as:

$$N_T = \sum_{t=1}^{\infty} t \times P \times P_e^{t-1}$$

This infinite series converges to :

$$N_T = 1/P \qquad (4.6)$$

Therefore, the throughput of pure ARQ with matrix retransmission is:

$$\eta_{A1} = P = (1-P_l)^{256} \qquad (4.7)$$

If only the lost cells are requested for retransmission, (second strategy), the throughput efficiency will be:

$$\eta_{A2} = 1 - P_l \qquad (4.8)$$

Throughput efficiencies η_{A1} and η_{A2} are plotted in Figure 4.5. The superiority of the second strategy (lost cells retransmission) over the first strategy (full matrix retransmission) is very clear. The first strategy provides a throughput of almost 100% up to BER=10^{-4}, while the throughput of the first strategy drops below 40% for the same BER.

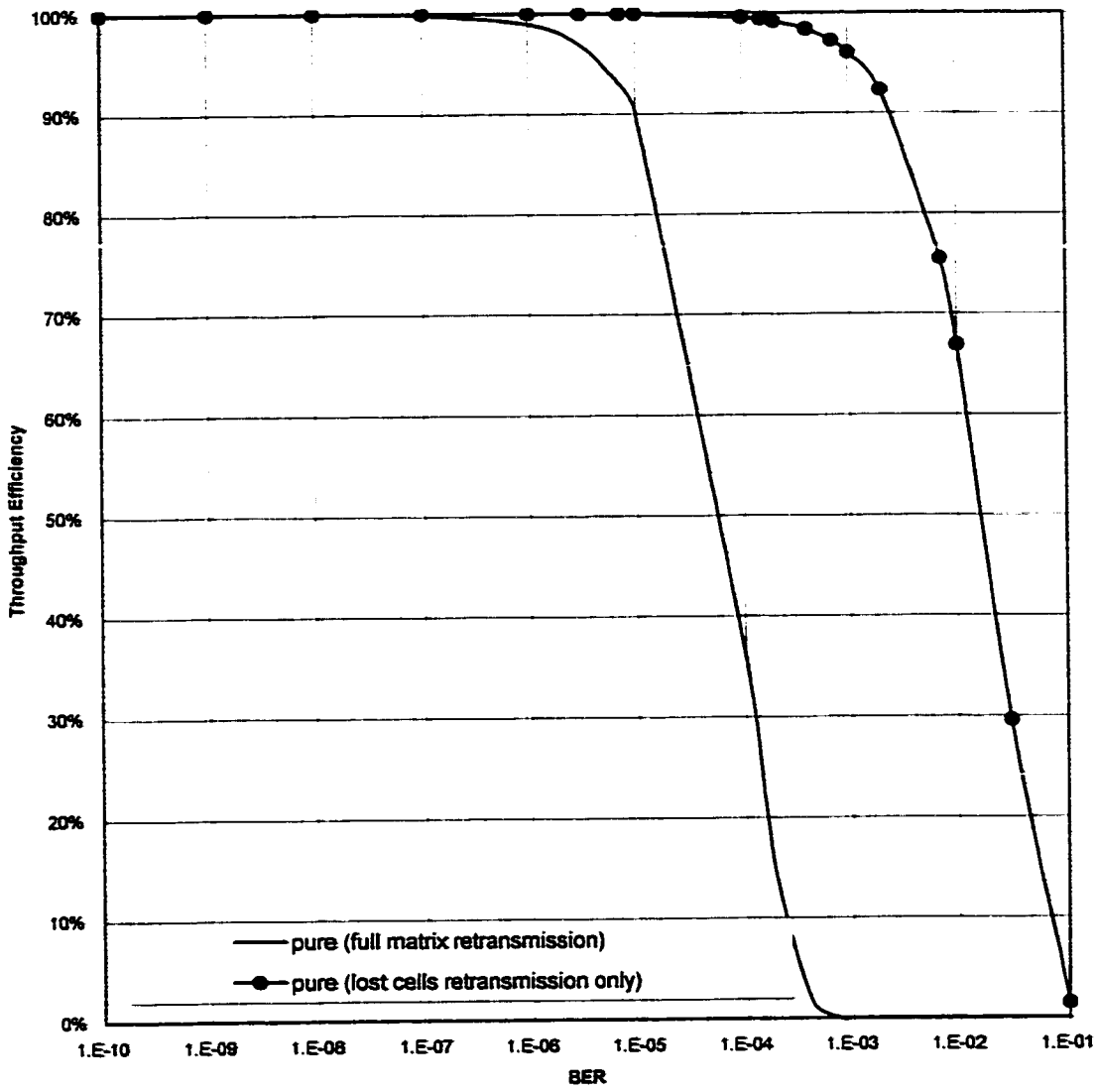


Figure 4.5 : Throughput efficiency for pure ARQ with full matrix retransmission and lost cells retransmission.

4.4 Hybrid- ARQ:

An ARQ scheme can be used in conjunction with the PCPC code. The resulting system is a *Hybrid-ARQ* system. In this system the PCPC attempts to recover lost cells, and any unrecovered cells are handled by the ARQ scheme. Both retransmission strategies: full matrix retransmission and lost cells retransmission will be analyzed.

Clearly, when PCPC is used alone, it provides a constant throughput = k/n which is the rate of the code. For the 16 × 16 PCPC, this constant throughput is 0.88. Hybrid-ARQ will maintain this throughput for good channels, but the throughput degrades for poor channels due to the increased retransmission requests. For hybrid-ARQ throughput efficiency, (4.4) can be rewritten as the following

$$\eta_H = \frac{0.88}{N_T} \qquad (4.9)$$

For the case when the full matrix is requested for retransmission and using (4.6) above:

$$\eta_{H,I} = 0.88\,P \qquad (4.10)$$

$$\text{where } P = 1 - \sum_{i=4}^{256} U_i P_l^i (1-P_l)^{256-i} \quad , \qquad (4.11)$$

where P_l is pre-decoding cell loss rate.

Comparison between the throughput efficiency of the pure ARQ and the hybrid-ARQ for the case of retranslating the full matrix is shown in Figure 4.6 . The PCPC has two opposing effects on the throughput of the hybrid system. On one hand it adds redundancy, thus lowering the throughput. On the other hand it reduces the number of retransmission, thus enhancing the throughput. The ultimate result of the throughput depends on which effect is dominant. For good channels (low BER), the matrices are received successfully from the first time and there is hardly any need for retransmission even for pure ARQ. Therefore pure ARQ, having no redundancy, yields a better performance. The hybrid ARQ becomes more advantageous at larger BERs where the reduction in the number of retransmissions pays off the added redundancy.

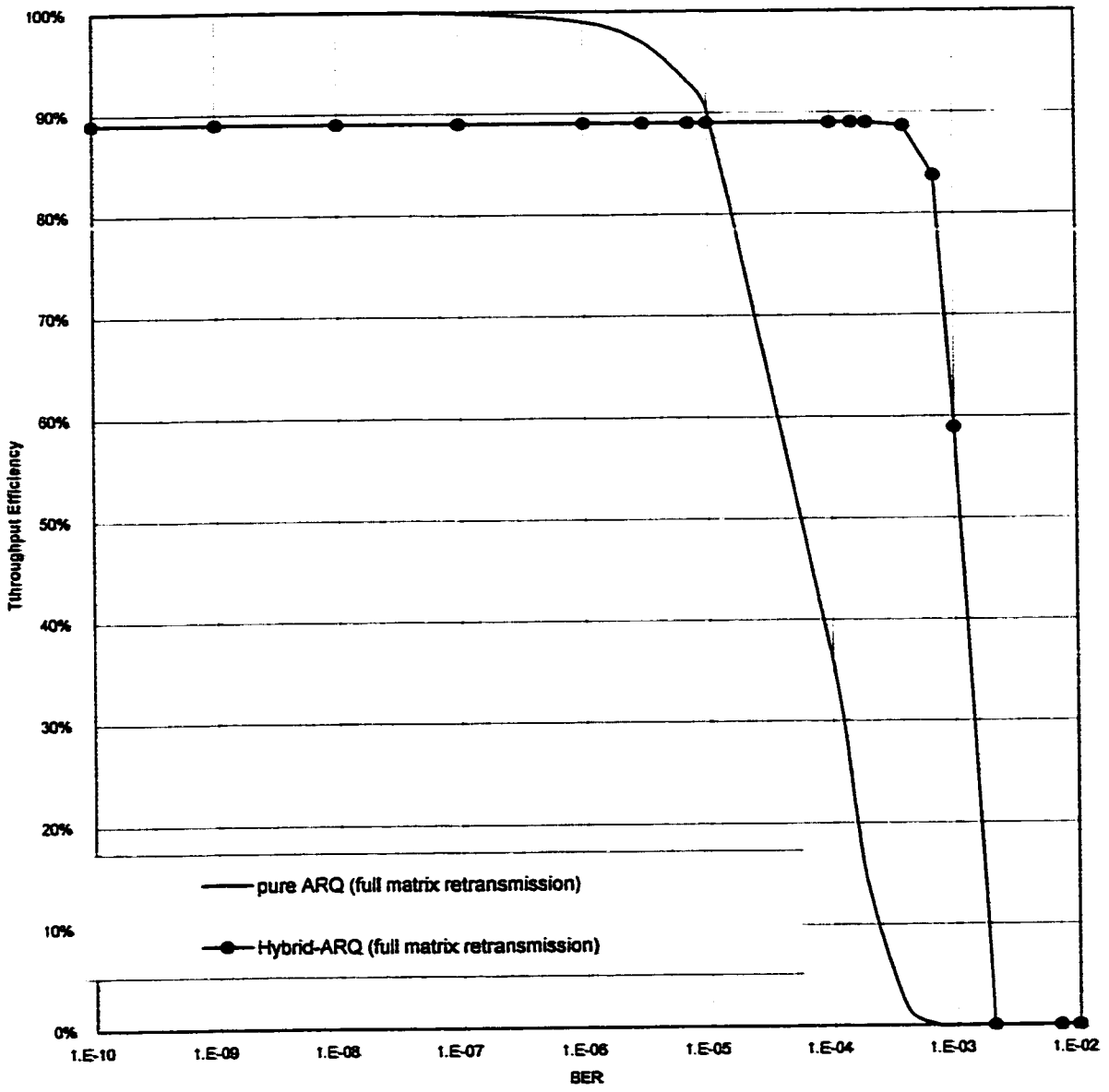


Figure 4.6 : Throughout efficiency for the pure ARQ and hybrid-ARQ using full matrix retransmission

If only the lost cells are requested for retransmission then the throughput efficiency of the hybrid ARQ system is given by:

$$\eta_{H2} = 0.88\, P_d \qquad (4.11)$$

where P_d is post decoding cell loss rate.

The comparison of the two retransmission strategies for hybrid ARQ is shown in Figure 4.7. Once again, the strategy of lost cells retransmission provides a better throughput. However it can be seen that the degree of improvement of the second strategy over the first strategy is less than that for pure ARQ (Fig. 4.5). This is because the probability of a matrix to be clean after decoding is significantly improved. For example, in pure ARQ any one lost cell would cause the matrix to be retransmitted. But for the hybrid scheme, at least four cells must remain unrecovered to cause matrix retransmission. And in fact, more than 90% of all the patterns consisting of up to 12 cells are recoverable.

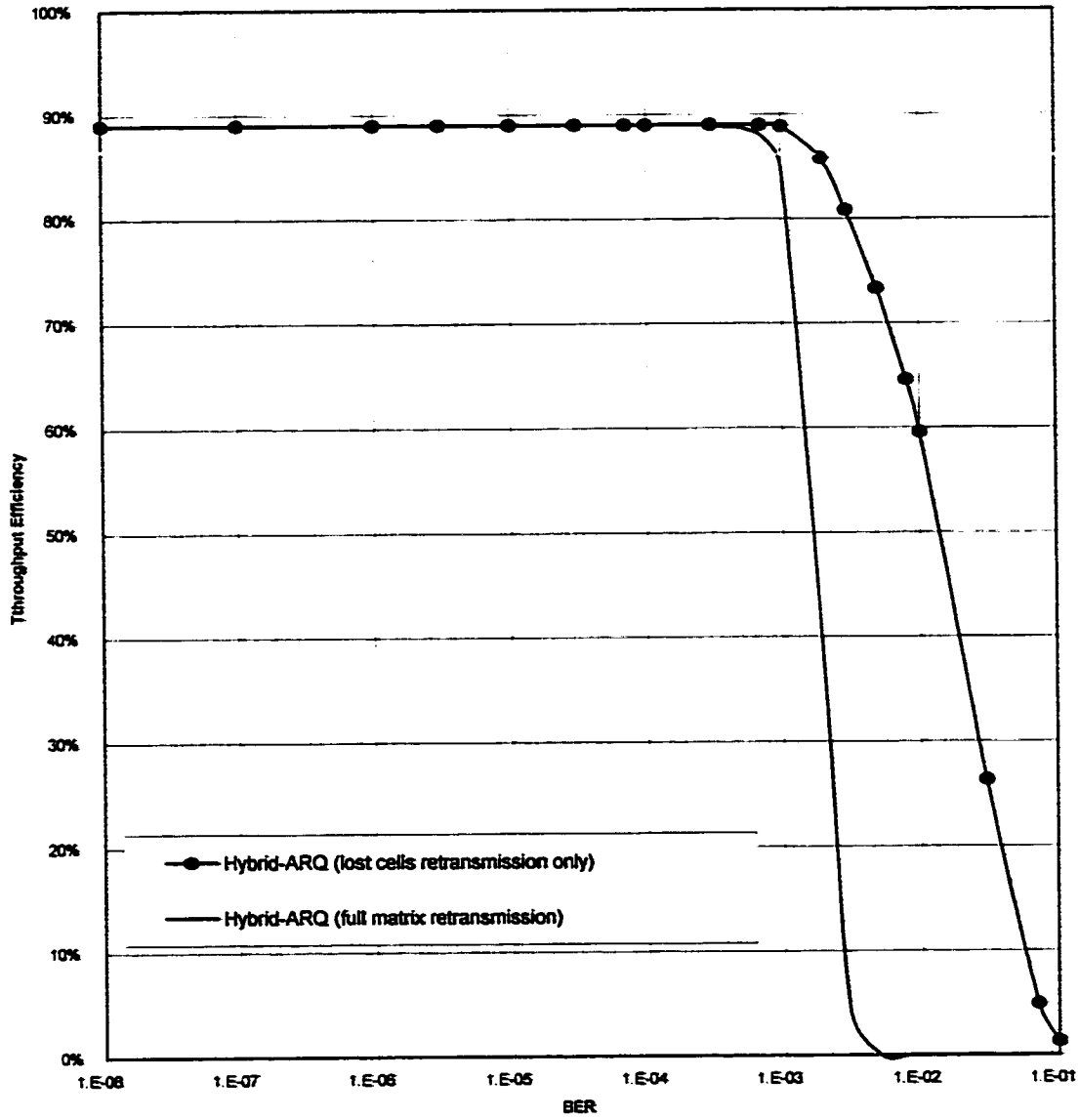


Figure 4.7: Hybrid ARQ throughput efficiency for both retransmission strategies (full matrix or lost cells only).

Finally, we compare the throughput of pure ARQ to hybrid ARQ for the second retransmission strategy. This is shown in Figure 4.8. Surprisingly, pure ARQ provides better throughput over all values of BER! This phenomenon can be explained with reference to the two opposing effects mentioned earlier. Although the PCPC reduces the post-decoding CLR significantly, which results reducing the number of retransmissions (note that the hybrid scheme maintains a fixed throughput over a wider range of BER), its effect is not enough to pay off the cost of increased redundancy. In other words the improvement in throughput due to reduced retransmissions was eaten up by the redundancy of the code.

It is important to mention here that this conclusion is valid under the assumption of infinite buffer of the SR protocol. If the buffer size is finite, both schemes will suffer from degradation of throughput due to buffer overflow. However as more cells are requested for retransmission in the pure ARQ scheme, it is expected to suffer more from the finite buffer size. Th net effect on the throughput needs to be studied.

4.5 Summery:

The PCPC was found to be very efficient in reducing ATM CLR for wide range of BER. If a feedback channel is available and an ARQ scheme is feasible, the hybrid ARQ/PCPC is still advantageous for the simple retransmission strategy of retransmitting the full matrix. If the more complex retransmission strategy of lost cells retransmitting can be implemented, then the pure ARQ scheme can handle the situation efficiently.

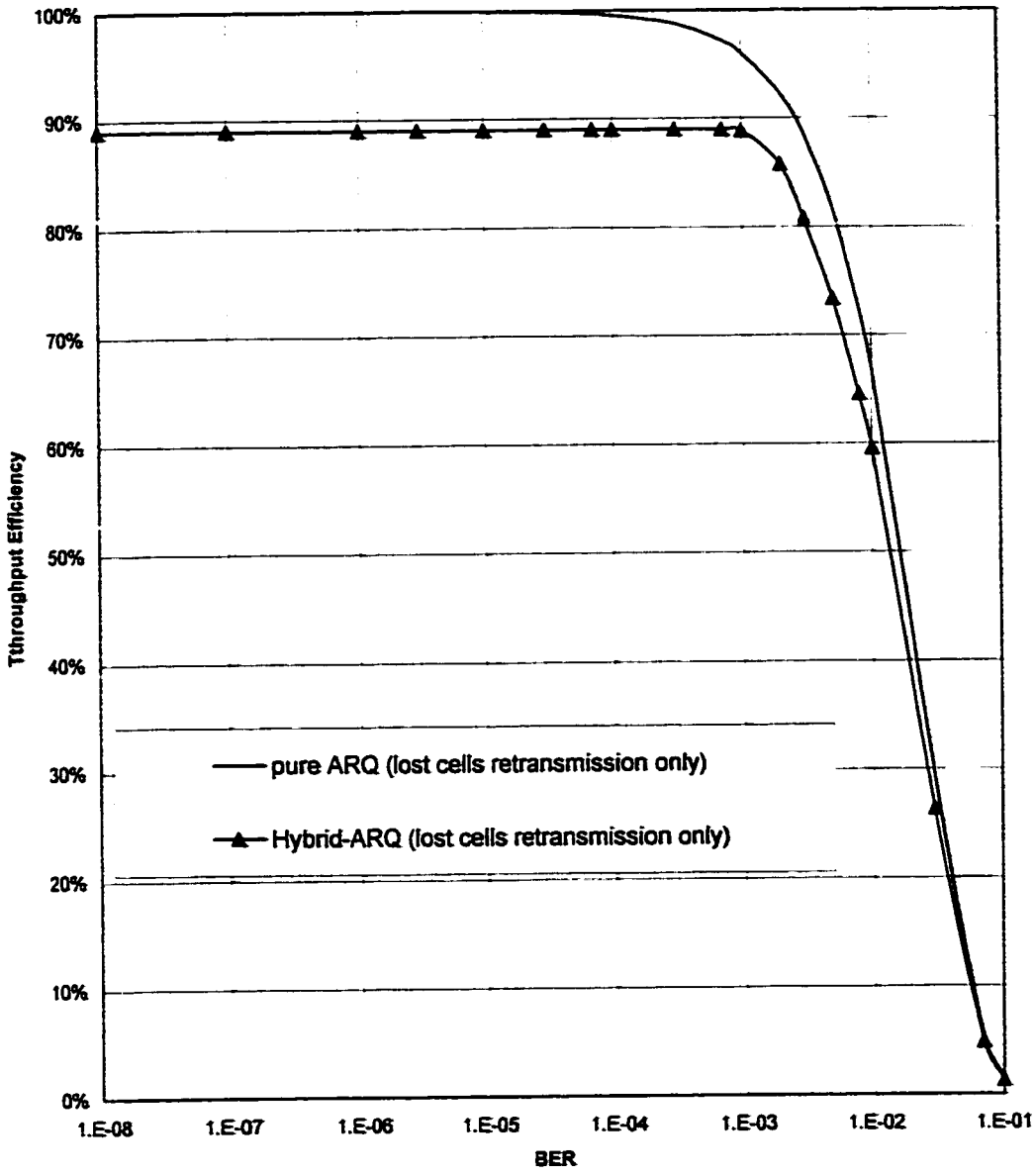


Figure 4.8 : The throughput efficiency for pure and hybrid ARQ when only the lost cells only are requested for retransmission.

CHAPTER 5

CONCLUSIONS AND SUGGESTIONS FOR FURTHER WORKS

A novel study about the structures of lost cell patterns in product codes is presented. This study can be useful for evaluating the performance of any product code. Then this study is used to evaluate the performance of the two dimensional parity check product code (PCPC). This study is partially used to derive a bound for the post decoding cell loss rate which is tighter than what is reported in the literature. This is done in a simple and computationally fast algorithm.

Using the same study in a relatively more complicated way leads to deriving a tight close-to-exact upper bound on the post decoding cell loss rate. A third bound was derived by considering only the unrecoverable cells after decoding instead of assuming that the whole pattern is unrecoverable. The results up to here are made general by finding the Post-decoding cell loss rate versus the

pre-decoding cell loss rate. The cells can be of any type and size and not necessarily ATM cells. The results showed that the cell loss rate is reduced considerably by using the PCPC. The accuracy of the results is checked by some simulation points. The code under study is compared with the one dimensional parity check code and an RS code used for erasure decoding . The PCPC showed a much better performance than both for good channels and comparable performance for bad channels.

Since one of the most serious problems in ATM networks is the cells discarding during the transmission, some error control techniques are studied for cell loss recovery for ATM networks. The PCPC is attractive due its extremely simple encoding and decoding (modulo-2). Thus it is applied for cell loss recovery in ATM networks. By assuming the cell loss took place only due error detection in the header, we evaluate the post decoding cell loss rate versus bit error rate.

Also we studied the effect of using an ARQ system to recover lost cells by requesting retransmission of those cells or the whole matrix. Also, we analyzed the performance of a hybrid ARQ/FEC coding scheme over ATM networks. For FEC we adopted the same simple product code which was analyzed throughout this thesis (the PCPC). On top of that a Selective Repeat ARQ (SR-ARQ) scheme is used resulting in a hybrid error control scheme.

As expected the hybrid-ARQ showed a better throughput than the pure ARQ if the full matrix is requested for retransmission for any lost cells. However, if the complex system of retransmission the lost cells only can be built, the pure ARQ will give excellent throughput.

Suggestions for further works:

In view of the findings of this work, we would like to make the following suggestions for carrying out further work in this area:

1. Utilizing the study of structural analysis of lost cell patterns in evaluating more complicated codes than PCPC.

2. Studying the code for ATM networks using a matrix larger than the 16×16 matrix used in this study. This requires introducing a more advanced method of cell loss detection.

3. In this study, it is assumed that all the cells which are not lost are error free. The erroneously received cells may lead to wrong lost cells recovery by the modulo-2 addition. The effect of this in the overall post decoding cell loss rate can be studied. Also the probability of receiving cells containing some errors can be studied.

4. There are many reasons for cell discarding in ATM networks. The most important reason among these, which was not considered in this thesis, is the buffer overflow. When the buffer is finite, considering buffer overflow is expected to make a clear difference in evaluating the post decoding cell loss rate versus bit error rate.

5. A stronger FEC code can be tried for hybrid-ARQ system for the purpose of error control in ARQ systems.

6. In the ARQ system considering a real feedback channel and a finite buffer size is another way to extend the work in this thesis.

References

[1] J. G. Proakis, *Digital Communications*, New York: McGraw-Hill, 1983.

[2] Simon Haykin, *Communication Systems*, 3'ed edition, Chap 12., New York: Joun and Wiley & Sons, 1994.

[3] Shu Lin and D. Castello, *Error Control Coding: Fundamentals and Applications*, Englewood Cliffs, N.J.: Prentice-Hall1983.

[4] S. B. Wicker, *Error Control Systems*, Englewood Cliffs, N.J.: Prentice-Hall Inc, 1995.

[5] Peter Sweeney, *Error Control Coding: An Introduction*. Englewood Cliffs, N.J: Prentice-Hall 1991.

[6] L. Hughes, *Introduction to Data communications*, Jones and Bartlett Publishers, 1996.

[7] Othmar Kayes, *ATM Networks*, second edition, International Thomson Publishing Inc. 1997.

[8] H.J.Dutton . P. Lenhard, *Asynchronous Transfer Mode (ATM)*, Englewood Cliffs, N.J.: Prentice-Hall Inc,1995.

[9] Martin. D. Prycker, *Asynchronous Transfer Mode*, Ellis horwood,1992.

[10] Uyless Black, *ATM Volume II*. Englewood Cliffs, N.J.: Prentice-Hall Inc,1995.

[11] Raif .O. Onvural, *Asynchronous Transfer Mode Networks*, Artech House Inc. 1995.

[12] Thomas. Chen and Stephen Liu, *ATM Switching Systems*, Artech House Inc, 1995.

[13] G.Kesidis, *ATM Network Performance*, Kluwer Academic Publishers, 1993.

[14] M. A, Kousa and A, H, Mugaibel, "Cell Loss Recovery Using Two-Dimensional Erasure Correction for ATM Networks", *7'th International Conference on Tele communication Systems*, March 1999, pp. 85-89.

[15] Hiroshi Ohta and Tokuhiro Kitami, "A Cell Loss Recovery Method using FEC in ATM Networks", *IEEE journal on selected areas in communication*, Vol. 9, December 91, pp. 1471-1483.

[16] A. K. Elhakeem, M. A. Kousa, and H. Yang, "A hybrid Multilayer Error Control Technique for multihop ATM Network", 1998, *IEEE international Conference on Communications (ICC 98)* Vol. 2, pp. 1168-1173.

[17] M. A. Kousa, A. K. Elhakeem and H. Yang, "Performance of ATM Networks Under Hybrid ARQ/FEC Error Control Scheme", *IEEE/ACM Transactions on Networking*, Vol.7, December 1999, pp. 917-925.

[18] Hyo T. Lim and Joo Song, "Cell Loss Recovery method in B-ISDN/ ATM networks." , *Electronics Letters* 25th May,95.Vol.31 No.11, pp. 849-851.

[19] Hyo T. Lim, D. Nyang and JooSeok Song, "Improving the performance of cell-loss recovery in ATM networks", *Electronics Letters* 15th August 1996, Vol. 32 No.17 pp. 1540-1542.

[20] J.Bibb Cin and Dennis N. Mcgregor , "A recommended Error Control Architecture For ATM Networks With Wireless Links", *IEEE Journal on Selected Areas In Communications*, Vol.5,No.1 , Jan 97, pp. 16-28.

[21] Hongyun Chen, Keno Takahahi, and Tapio Erke, "The Analysis of The Application of ARQ Mechanism in ATM Network", *IEE International Conference on Communication* 1992, Vol .1, pp. 41-44.

[22] M.chiani and A. volta, "Hybrid ARQ /FEC Techniques for wireless ATM local area networks", *7'th IEEE int. symp. on personal, indoor and mobile radio comm;* 96, vol3, pp. 898-902.

[23] Dvid Moore and Michael Rice "Variable Rate Error Control For Wireless ATM Netwoks", IEE *International Conference on Communication* 1995 Vol. 2, pp 988-992.

[24] Yohannes A. tesfai and G. Willson, "FEC schemes for ATM traffic over wireless links", *Proceeding IEEE military communication conference* 1996 (milcom'96), Vol. 3, pp. 948-953.

[25] Kenji Kawahara, and Yuji Oie, "Forward Error Correction in ATM Networks" *Proceedings of IEEE networking for Global Communication,* INFOCOM'94, Vol. 3, pp. 1150-1159.

[26] Peter R, Denz Arnc A, Nilsson, "Performance of Error Control Coding Techniques for Wireless ATM" *1998 IEEE International Conference on Communications (ICC 98)*, Vol. 2, pp. 1099-1103.

[27] Walaa A. Hamouda and Peter J. Mclane, "Performance of ATM Cell Transmission Via Regenerative Satellite Links", *1998 IEEE international Conference on Communications (ICC 98)* Vol. 3, pp. 1436-1442.

[28] Q. Ding and He Lin, "Throughput Analysis of a Hybrid Error Control with Early-Stop ARQ Scheme for Wireless ATM", *1999 IEEE International Conference on Communications; ICC'99*, Vol. 3, pp. 1859-1863.

[29] U Varshney, "Error Control Techniques for ATM Networks", *1999 IEEE International Conference Performance, Computing and Communications;* pp. 104-110.

[30] C. H. Ng, L. Zhang, T. Cheng, C. Ten, "Cell Loss Probability of a Finite ATM Buffer Queue", *IEE Proceedings-Communications*; Feb. 99; pp. 9-14.

APPENDIX A

A.1 Generating the Recoverable Basic Patterns (Algorithmic Approach):

The Systematic procedure of producing the recoverable basic patterns:

1. For *i* cells the first recoverable basic pattern is i occupied columns with one cell in each. If *N*< *i*, the first row of the matrix is filled and the extra cells are placed at the first column.

2. The next recoverable basic patterns is produced by moving the most right cell in the previous pattern to the first column, as shown:

3. The following recoverable basic patterns are produced by repeating the second step until all the cells are moved to the first column or the first column is fully occupied.

The above procedure can be translated to the following algorithm:

```
Do j= 0 to lostcells
(unless Clm1_cellls = N)

            If M≥ lostcells

        Ocpd_colms=lostcells-j
              Clm1_cellls=I+j
           Else

              Ocpd_colms=M-j
              Clm1_cellls=j+( lostcells-M)+j
Continue j.
```

A.2 Generating the Bodies: Algorithmic Approach

As it was explained in Chapter 2, finding all the bodies is the first step in building all the unrecoverable patterns. To find all the bodies it will be helpful to classify them into different types according to the number of different column sizes. We define *type-m* body as a body with *m* different sizes of columns. That means *type-1* bodies have only one size of columns while *type-2* bodies have two different sizes of columns. Clearly, the first two columns of any body must be of the same size regardless of the size.

The following table shows the smallest body of each body type:

Type name	The minimum size body	Shape of the body
type-1	4 cells.	Xx xx
type-2	8 cells.	xxx xxx xx
type-3	13 cells.	xxxx xxxx xxx xx
type-4	19 cells.	xxxxx xxxxx xxxx xxx xx
type-5	26 cells.	xxxxxx xxxxxx xxxxx xxxx xxx xx
type-6	34 cells.	xxxxxxx xxxxxxx xxxxxx xxxxx xxxx xxx xx

Table A. 1: Smallest body of each type.

Type-1 bodies:

Finding all the type-1 bodies is a simple task. In the same time, fortunately enough, the recoverable patterns generated from most of them are huge and easily segregated from the unrecoverable ones. All the columns of any type-1 body have the same number of cells (same column size). For any number of cells the minimum type-1body is always the 2×2 body.

A type-1 body takes the form of an $s \times t$ matrix. For i lost cells s can be take any value between 2 and $\left\lfloor \frac{i}{2} \right\rfloor$, and for any s, t can take any value between 2 and $\left\lfloor \frac{i}{s} \right\rfloor$.

The total number of type-1 bodies is:

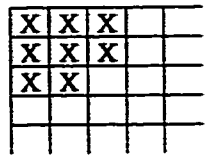
$$B_{tl} = \sum_{s=2}^{\left\lfloor \frac{i}{2} \right\rfloor} \sum_{t}^{\left\lfloor \frac{s}{2} \right\rfloor} 1$$

After some manipulation, it can be written as:

$$B_{tl} = 1 + \sum_{s=3}^{\left\lfloor \frac{i}{2} \right\rfloor} \left\lfloor \frac{s}{2} \right\rfloor$$

Type-2 bodies:

The smallest body size of type-2 is the following:



That means that type-2 patterns are available only for number of cells ≥ 8.

Any type-2 body can be classified into two blocks corresponding to the two sizes of columns. let us call these two blocks to be $s \times t$ and $u \times v$. The number of type-2 bodies is:

$$B_{t2} = \sum_{s=3}^{a} \sum_{t=2}^{b} \sum_{u=2}^{c} \sum_{v=1}^{d} 1$$

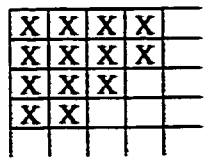
where $a = \left\lfloor \frac{i-2}{2} \right\rfloor$, $b = \left\lfloor \frac{i-2}{s} \right\rfloor$, $c = \min[(s-1),(i - s \times t)]$ and $d = \left\lfloor \frac{i - s \times t}{u} \right\rfloor$

Clearly, this can be rewritten as:

$$B_{t2} = \sum_{s=3}^{a} \sum_{t=2}^{b} \sum_{u=2}^{c} \left\lfloor \frac{i - s \times t}{u} \right\rfloor$$

Type-3 bodies:

The smallest body size of type-2 is the following:



That means that type-3 patterns can not appear for number of cells < 13.

let us call the three different block sizes of any type-3 body to be $s\times t$, $u\times v$ and $j\times k$. By studying the minimum and maximum of each block, the number of type-3 bodies can be found as the following:

$$B_{t3}=\sum_{s=4}^{a}\sum_{t=2}^{b}\sum_{u=3}^{c}\sum_{v=1}^{d}\sum_{j=2}^{e}\sum_{k=1}^{f}1$$

where $a=\left\lfloor\frac{i-5}{2}\right\rfloor$, $b=\left\lfloor\frac{i-5}{s}\right\rfloor$, $c=\min[(s-1),(i-s\times t)]$, $d=\left\lfloor\frac{i-s\times t-2}{u}\right\rfloor$

$e=\min[(u-1),(i-s\times t-u\times v)]$ $f=\left\lfloor\frac{i-s\times t-u\times v}{j}\right\rfloor$

Clearly, this can be rewritten as:

$$B_{t3}=\sum_{s=4}^{a}\sum_{t=2}^{b}\sum_{u=3}^{c}\sum_{v=1}^{d}\sum_{j=2}^{e}\left\lfloor\frac{i-s\times t-u\times v}{j}\right\rfloor$$

Higher Type bodies:

As it was shown in the Table A.1 the smallest body of type-4 has 19 cells. That means that up to eighteen lost cells, the bodies of the first three types are enough to create all the UBPs. For 19 and above cells, Table A.2 shows the percentage of the patterns generated from the first three types for different sizes of matrixes.

	8×8 Matrix	16×16 Matrix	32×32 Matrix
19	100.0%	100.0%	100.0%
20	99.9%	100.0%	100.0%
21	99.3%	100.0%	100.0%
22	97.8%	100.0%	100.0%
23	95.0%	100.0%	100.0%
24	90.1%	100.0%	100.0%
25	83.6%	100.0%	100.0%
26	74.8%	99.9%	100.0%
27	65.6%	99.8%	100.0%
28	56.4%	99.5%	100.0%
29	48.9%	99.1%	100.0%

Table A. 2: Percentage of the patterns generated from the first three types for different sizes of matrixes.

It is clear from Table A.2 that when the matrix size increase the patterns generated from higher types of bodies are negligible. Moreover for small matrixes although these patterns are not negligible, the are most likely unrecoverable. As an example, Table A.2 shows that the 8×8 matrix has some patterns from higher types bodies, but all of the are unrecoverable because they contain > 16 cell.

The above discussion leads to a general conclusion that the first three types are enough to build all the UBP's and consequently generate all the possible patterns for any number of cells up to 18. In the same time they are able to build most of the patterns for higher number of cells. Any higher type pattern can be safely assumed unrecoverable without knowing its shape.

In the following three table the bodies of the first three types are shown and the number of UBP's the can be built using that body for different number of cells.

Body #	Number of lost cells	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	XX XX	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	XX XX			1	2	3	4	5	6	7	8	9	10	11	12	13
3	XX XX					1	2	3	4	5	6	7	8	9	10	11
4	XX XX XX							1	2	3	4	5	6	7	8	9
5	XX XX XX XX									1	2	3	4	5	6	7
6	XX XX XX XX XX											1	2	3	4	5
7	XX XX XX XX XX XX													1	2	3
8	XX XX XX XX XX XX XX															1
9	XXX XXX			1	2	3	4	5	6	7	8	9	10	11	12	13
10	XXX XXX						1	2	3	4	5	6	7	8	9	10
11	XXX XXX XXX									1	2	3	4	5	6	7
12	XXX XXX XXX XXX												1	2	3	4
13	XXX XXX XXX XXX XXX															1
14	XXXX XXXX					1	2	3	4	5	6	7	8	9	10	11
15	XXXX XXXX XXXX									1	2	3	4	5	6	7
16	XXXX XXXX XXXX XXXX													1	2	3
17	XXXXX XXXXX							1	2	3	4	5	6	7	8	9
18	XXXXX XXXXX XXXXX												1	2	3	4
19	XXXXXX XXXXXX									1	2	3	4	5	6	7
20	XXXXXX XXXXXX XXXXXX														1	2
21	XXXXXXX XXXXXXX											1	2	3	4	5
22	XXXXXXXX XXXXXXXX													1	2	3
23	XXXXXXXXX XXXXXXXXX															1
	Total Bodies			3	3	5	6	8	8	12	12	14	16	19	20	23
	Total UBP's			5	8	13	19	27	35	47	59	73	89	108	128	151

Table A. 3: Type-1 Bodies and number of UBPs for each body.

Body #	Number of lost cells	8	9	10	11	12	13	14	15	16	17	18
1		1	2	3	4	5	6	7	8	9	10	11
2				1	2	3	4	5	6	7	8	9
3					1	2	3	4	5	6	7	8
4						1	2	3	4	5	6	7
5							1	2	3	4	5	6
6								1	2	3	4	5
7								1	2	3	4	5
8									1	2	3	4
9										1	2	3
10											1	2
11										1	2	3
12											1	2
13												1
14												1
15				1	2	3	4	5	6	7	8	9
16					1	2	3	4	5	6	7	8
17						1	2	3	4	5	6	7
18								1	2	3	4	5
19								1	2	3	4	5
20											1	2
21								1	2	3	4	5
22										1	2	3

23												1
24											1	2
25												1
26										1	2	3
27												1
28												1
29						1	2	3	4	5	6	7
30							1	2	3	4	5	6
31								1	2	3	4	5
32								1	2	3	4	5
33										1	2	3
34												1
35											1	2
36												1
37										1	2	3
38												1
39								1	2	3	4	5
40									1	2	3	4
41										1	2	3
42												2
43										1	2	3
44												1
45												1
46										1	2	3
47											1	2
48												1
49												1

50												1
	Total Bodies	1	1	3	5	8	10	18	20	29	35	50
	Total UBPs	1	2	5	10	18	28	46	66	95	130	182

Table A. 4: Type-2 Bodies and number of UBPs for each body:

Body #	*Number of last cells*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*
1									1	2	3	4	5	6
2											1	2	3	4
3													1	2
4												1	2	3
5														1
6													1	2
7											1	2	3	4
8													1	2
9														1
10												1	2	3
11														1
12													1	2
13													1	2
14														1
	Total Bodies								1	1	3	5	10	14
	Total UBPs								1	2	5	10	20	34

Table A. 5 Type-3 Bodies and number of UBPs for each body:

The following tables is a summery of the above three tables:

Number of lost cells	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*
Bodies of Typ-1	1	1	3	3	5	6	8	8	12	12	14	16	19	20	23
Bodies of Typ-2					1	1	3	5	8	10	18	20	29	35	50
Bodies of Typ-3										1	1	3	5	10	14
Total Bodies	1	1	3	3	6	7	11	13	20	23	33	39	53	65	87

Table A. 6 Summery of number of bodies for each number of cells.

Number of lost cells	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*
UBPs of Bodies of Type-1	1	2	5	8	13	19	27	35	47	59	73	89	108	128	151
UBPs of Bodies of Type-2					1	2	5	10	18	28	46	66	95	130	182
UBPs of Bodies of Type-3										1	2	5	10	20	34
Total UBPs	1	2	5	8	14	21	32	45	65	88	121	160	213	278	367

Table A. 7 Summery of number of UBPs for each number of cells.